

activemq-cpp-3.2.5

Generated by Doxygen 1.7.4

Sun Jan 8 2012 23:14:17



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>25</b>
3.1	Data Structures . . . . .	25
<b>4</b>	<b>File Index</b>	<b>67</b>
4.1	File List . . . . .	67
<b>5</b>	<b>Namespace Documentation</b>	<b>93</b>
5.1	activemq Namespace Reference . . . . .	93
5.1.1	Detailed Description . . . . .	93
5.2	activemq::cmsutil Namespace Reference . . . . .	94
5.3	activemq::commands Namespace Reference . . . . .	95
5.4	activemq::core Namespace Reference . . . . .	96
5.5	activemq::core::policies Namespace Reference . . . . .	97
5.6	activemq::exceptions Namespace Reference . . . . .	97
5.7	activemq::io Namespace Reference . . . . .	98
5.8	activemq::library Namespace Reference . . . . .	98
5.9	activemq::state Namespace Reference . . . . .	98
5.10	activemq::threads Namespace Reference . . . . .	98
5.11	activemq::transport Namespace Reference . . . . .	99
5.12	activemq::transport::correlator Namespace Reference . . . . .	100
5.13	activemq::transport::failover Namespace Reference . . . . .	100

5.14	activemq::transport::inactivity Namespace Reference	100
5.15	activemq::transport::logging Namespace Reference	100
5.16	activemq::transport::mock Namespace Reference	101
5.17	activemq::transport::tcp Namespace Reference	101
5.18	activemq::util Namespace Reference	101
5.19	activemq::wireformat Namespace Reference	102
5.20	activemq::wireformat::openwire Namespace Reference	102
5.21	activemq::wireformat::openwire::marshal Namespace Reference	103
5.22	activemq::wireformat::openwire::marshal::v1 Namespace Reference	103
5.23	activemq::wireformat::openwire::marshal::v2 Namespace Reference	106
5.24	activemq::wireformat::openwire::marshal::v3 Namespace Reference	110
5.25	activemq::wireformat::openwire::marshal::v4 Namespace Reference	113
5.26	activemq::wireformat::openwire::marshal::v5 Namespace Reference	116
5.27	activemq::wireformat::openwire::marshal::v6 Namespace Reference	119
5.28	activemq::wireformat::openwire::utils Namespace Reference	122
5.29	activemq::wireformat::stomp Namespace Reference	122
5.30	cms Namespace Reference	122
5.30.1	Detailed Description	125
5.31	decaf Namespace Reference	125
5.31.1	Detailed Description	125
5.32	decaf::internal Namespace Reference	125
5.33	decaf::internal::io Namespace Reference	126
5.34	decaf::internal::net Namespace Reference	126
5.35	decaf::internal::net::ssl Namespace Reference	127
5.36	decaf::internal::net::ssl::openssl Namespace Reference	127
5.37	decaf::internal::net::tcp Namespace Reference	128
5.38	decaf::internal::nio Namespace Reference	128
5.39	decaf::internal::security Namespace Reference	129
5.40	decaf::internal::util Namespace Reference	129
5.41	decaf::internal::util::concurrent Namespace Reference	129
5.42	decaf::io Namespace Reference	130
5.43	decaf::lang Namespace Reference	131
5.43.1	Function Documentation	133
5.43.1.1	operator!=	133



5.43.1.2	operator!=	133
5.43.1.3	operator!=	133
5.43.1.4	operator!=	133
5.43.1.5	operator==	133
5.43.1.6	operator==	134
5.43.1.7	operator==	134
5.43.1.8	operator==	134
5.44	decaf::lang::exceptions Namespace Reference	134
5.45	decaf::net Namespace Reference	134
5.46	decaf::net::ssl Namespace Reference	136
5.47	decaf::nio Namespace Reference	136
5.48	decaf::security Namespace Reference	137
5.49	decaf::security::auth Namespace Reference	137
5.50	decaf::security::auth::x500 Namespace Reference	137
5.51	decaf::security::cert Namespace Reference	138
5.52	decaf::util Namespace Reference	138
5.53	decaf::util::comparators Namespace Reference	140
5.54	decaf::util::concurrent Namespace Reference	140
5.55	decaf::util::concurrent::atomic Namespace Reference	141
5.56	decaf::util::concurrent::locks Namespace Reference	142
5.57	decaf::util::logging Namespace Reference	142
5.57.1	Enumeration Type Documentation	143
5.57.1.1	Levels	143
5.58	decaf::util::zip Namespace Reference	144
5.59	std Namespace Reference	145
<b>6</b>	<b>Data Structure Documentation</b>	<b>147</b>
6.1	decaf::util::AbstractCollection< E > Class Template Reference	147
6.1.1	Detailed Description	149
6.1.2	Constructor & Destructor Documentation	149
6.1.2.1	AbstractCollection	149
6.1.2.2	~AbstractCollection	149
6.1.3	Member Function Documentation	149
6.1.3.1	add	150

6.1.3.2	addAll	150
6.1.3.3	clear	151
6.1.3.4	contains	152
6.1.3.5	containsAll	153
6.1.3.6	copy	153
6.1.3.7	equals	153
6.1.3.8	isEmpty	154
6.1.3.9	lock	154
6.1.3.10	notify	155
6.1.3.11	notifyAll	155
6.1.3.12	operator=	155
6.1.3.13	remove	156
6.1.3.14	removeAll	157
6.1.3.15	retainAll	157
6.1.3.16	toArray	158
6.1.3.17	tryLock	158
6.1.3.18	unlock	159
6.1.3.19	wait	159
6.1.3.20	wait	159
6.1.3.21	wait	160
6.1.4	Field Documentation	160
6.1.4.1	mutex	161
6.2	decaf::util::AbstractList< E > Class Template Reference	161
6.2.1	Detailed Description	161
6.2.2	Constructor & Destructor Documentation	162
6.2.2.1	~AbstractList	162
6.3	decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference	162
6.3.1	Detailed Description	162
6.3.2	Constructor & Destructor Documentation	163
6.3.2.1	~AbstractMap	163
6.4	decaf::util::AbstractQueue< E > Class Template Reference	163
6.4.1	Detailed Description	164
6.4.2	Constructor & Destructor Documentation	164

6.4.2.1	AbstractQueue . . . . .	164
6.4.2.2	~AbstractQueue . . . . .	164
6.4.3	Member Function Documentation . . . . .	164
6.4.3.1	add . . . . .	165
6.4.3.2	addAll . . . . .	165
6.4.3.3	clear . . . . .	166
6.4.3.4	element . . . . .	166
6.4.3.5	remove . . . . .	167
6.5	decaf::util::AbstractSequentialList< E > Class Template Reference . . .	167
6.5.1	Detailed Description . . . . .	167
6.5.2	Constructor & Destructor Documentation . . . . .	168
6.5.2.1	~AbstractSequentialList . . . . .	168
6.6	decaf::util::AbstractSet< E > Class Template Reference . . . . .	168
6.6.1	Detailed Description . . . . .	169
6.6.2	Constructor & Destructor Documentation . . . . .	169
6.6.2.1	~AbstractSet . . . . .	169
6.6.3	Member Function Documentation . . . . .	169
6.6.3.1	removeAll . . . . .	169
6.7	activemq::transport::AbstractTransportFactory Class Reference . . . . .	170
6.7.1	Detailed Description . . . . .	171
6.7.2	Constructor & Destructor Documentation . . . . .	171
6.7.2.1	~AbstractTransportFactory . . . . .	171
6.7.3	Member Function Documentation . . . . .	171
6.7.3.1	createWireFormat . . . . .	171
6.8	activemq::core::ActiveMQAckHandler Class Reference . . . . .	171
6.8.1	Detailed Description . . . . .	172
6.8.2	Constructor & Destructor Documentation . . . . .	172
6.8.2.1	~ActiveMQAckHandler . . . . .	172
6.8.3	Member Function Documentation . . . . .	172
6.8.3.1	acknowledgeMessage . . . . .	172
6.9	activemq::commands::ActiveMQBlobMessage Class Reference . . . . .	172
6.9.1	Constructor & Destructor Documentation . . . . .	174
6.9.1.1	ActiveMQBlobMessage . . . . .	174
6.9.1.2	~ActiveMQBlobMessage . . . . .	174

6.9.2	Member Function Documentation . . . . .	174
6.9.2.1	clone . . . . .	174
6.9.2.2	cloneDataStructure . . . . .	174
6.9.2.3	copyDataStructure . . . . .	174
6.9.2.4	equals . . . . .	175
6.9.2.5	getDataStructureType . . . . .	175
6.9.2.6	getMimeType . . . . .	175
6.9.2.7	getName . . . . .	175
6.9.2.8	getRemoteBlobUrl . . . . .	176
6.9.2.9	isDeletedByBroker . . . . .	176
6.9.2.10	setDeletedByBroker . . . . .	176
6.9.2.11	setMimeType . . . . .	176
6.9.2.12	setName . . . . .	176
6.9.2.13	setRemoteBlobUrl . . . . .	177
6.9.2.14	toString . . . . .	177
6.9.3	Field Documentation . . . . .	177
6.9.3.1	BINARY_MIME_TYPE . . . . .	177
6.9.3.2	ID_ACTIVEMQBLOBMESSAGE . . . . .	177
6.10	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference . . . . .	177
6.10.1	Detailed Description . . . . .	178
6.10.2	Constructor & Destructor Documentation . . . . .	178
6.10.2.1	ActiveMQBlobMessageMarshaller . . . . .	178
6.10.2.2	~ActiveMQBlobMessageMarshaller . . . . .	178
6.10.3	Member Function Documentation . . . . .	178
6.10.3.1	createObject . . . . .	179
6.10.3.2	getDataStructureType . . . . .	179
6.10.3.3	looseMarshal . . . . .	179
6.10.3.4	looseUnmarshal . . . . .	180
6.10.3.5	tightMarshal1 . . . . .	180
6.10.3.6	tightMarshal2 . . . . .	181
6.10.3.7	tightUnmarshal . . . . .	181
6.11	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller Class Reference . . . . .	182

6.11.1	Detailed Description . . . . .	182
6.11.2	Constructor & Destructor Documentation . . . . .	183
6.11.2.1	ActiveMQBlobMessageMarshaller . . . . .	183
6.11.2.2	~ActiveMQBlobMessageMarshaller . . . . .	183
6.11.3	Member Function Documentation . . . . .	183
6.11.3.1	createObject . . . . .	183
6.11.3.2	getDataStructureType . . . . .	183
6.11.3.3	looseMarshal . . . . .	183
6.11.3.4	looseUnmarshal . . . . .	184
6.11.3.5	tightMarshal1 . . . . .	184
6.11.3.6	tightMarshal2 . . . . .	185
6.11.3.7	tightUnmarshal . . . . .	185
6.12	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller Class Reference . . . . .	186
6.12.1	Detailed Description . . . . .	186
6.12.2	Constructor & Destructor Documentation . . . . .	187
6.12.2.1	ActiveMQBlobMessageMarshaller . . . . .	187
6.12.2.2	~ActiveMQBlobMessageMarshaller . . . . .	187
6.12.3	Member Function Documentation . . . . .	187
6.12.3.1	createObject . . . . .	187
6.12.3.2	getDataStructureType . . . . .	187
6.12.3.3	looseMarshal . . . . .	187
6.12.3.4	looseUnmarshal . . . . .	188
6.12.3.5	tightMarshal1 . . . . .	188
6.12.3.6	tightMarshal2 . . . . .	189
6.12.3.7	tightUnmarshal . . . . .	189
6.13	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller Class Reference . . . . .	190
6.13.1	Detailed Description . . . . .	190
6.13.2	Constructor & Destructor Documentation . . . . .	191
6.13.2.1	ActiveMQBlobMessageMarshaller . . . . .	191
6.13.2.2	~ActiveMQBlobMessageMarshaller . . . . .	191
6.13.3	Member Function Documentation . . . . .	191
6.13.3.1	createObject . . . . .	191

6.13.3.2	getDataStructureType . . . . .	191
6.13.3.3	looseMarshal . . . . .	191
6.13.3.4	looseUnmarshal . . . . .	192
6.13.3.5	tightMarshal1 . . . . .	192
6.13.3.6	tightMarshal2 . . . . .	193
6.13.3.7	tightUnmarshal . . . . .	193
6.14	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller Class Reference . . . . .	194
6.14.1	Detailed Description . . . . .	194
6.14.2	Constructor & Destructor Documentation . . . . .	195
6.14.2.1	ActiveMQBlobMessageMarshaller . . . . .	195
6.14.2.2	~ActiveMQBlobMessageMarshaller . . . . .	195
6.14.3	Member Function Documentation . . . . .	195
6.14.3.1	createObject . . . . .	195
6.14.3.2	getDataStructureType . . . . .	195
6.14.3.3	looseMarshal . . . . .	195
6.14.3.4	looseUnmarshal . . . . .	196
6.14.3.5	tightMarshal1 . . . . .	196
6.14.3.6	tightMarshal2 . . . . .	197
6.14.3.7	tightUnmarshal . . . . .	197
6.15	activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller Class Reference . . . . .	198
6.15.1	Detailed Description . . . . .	198
6.15.2	Constructor & Destructor Documentation . . . . .	199
6.15.2.1	ActiveMQBlobMessageMarshaller . . . . .	199
6.15.2.2	~ActiveMQBlobMessageMarshaller . . . . .	199
6.15.3	Member Function Documentation . . . . .	199
6.15.3.1	createObject . . . . .	199
6.15.3.2	getDataStructureType . . . . .	199
6.15.3.3	looseMarshal . . . . .	199
6.15.3.4	looseUnmarshal . . . . .	200
6.15.3.5	tightMarshal1 . . . . .	200
6.15.3.6	tightMarshal2 . . . . .	201
6.15.3.7	tightUnmarshal . . . . .	201

6.16	activemq::commands::ActiveMQBytesMessage Class Reference . . . .	202
6.16.1	Constructor & Destructor Documentation . . . . .	204
6.16.1.1	ActiveMQBytesMessage . . . . .	204
6.16.1.2	~ActiveMQBytesMessage . . . . .	205
6.16.2	Member Function Documentation . . . . .	205
6.16.2.1	clearBody . . . . .	205
6.16.2.2	clone . . . . .	205
6.16.2.3	cloneDataStructure . . . . .	205
6.16.2.4	copyDataStructure . . . . .	205
6.16.2.5	equals . . . . .	206
6.16.2.6	getBodyBytes . . . . .	206
6.16.2.7	getBodyLength . . . . .	206
6.16.2.8	getDataStructureType . . . . .	207
6.16.2.9	onSend . . . . .	207
6.16.2.10	readBoolean . . . . .	207
6.16.2.11	readByte . . . . .	208
6.16.2.12	readBytes . . . . .	208
6.16.2.13	readBytes . . . . .	209
6.16.2.14	readChar . . . . .	210
6.16.2.15	readDouble . . . . .	210
6.16.2.16	readFloat . . . . .	210
6.16.2.17	readInt . . . . .	211
6.16.2.18	readLong . . . . .	211
6.16.2.19	readShort . . . . .	212
6.16.2.20	readString . . . . .	212
6.16.2.21	readUnsignedShort . . . . .	213
6.16.2.22	readUTF . . . . .	213
6.16.2.23	reset . . . . .	214
6.16.2.24	setBodyBytes . . . . .	214
6.16.2.25	toString . . . . .	214
6.16.2.26	writeBoolean . . . . .	215
6.16.2.27	writeByte . . . . .	215
6.16.2.28	writeBytes . . . . .	215
6.16.2.29	writeBytes . . . . .	216

6.16.2.30	writeChar . . . . .	216
6.16.2.31	writeDouble . . . . .	217
6.16.2.32	writeFloat . . . . .	217
6.16.2.33	writeInt . . . . .	218
6.16.2.34	writeLong . . . . .	218
6.16.2.35	writeShort . . . . .	218
6.16.2.36	writeString . . . . .	219
6.16.2.37	writeUnsignedShort . . . . .	219
6.16.2.38	writeUTF . . . . .	220
6.16.3	Field Documentation . . . . .	220
6.16.3.1	ID_ACTIVEMQBYTESMESSAGE . . . . .	220
6.17	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	220
6.17.1	Detailed Description . . . . .	221
6.17.2	Constructor & Destructor Documentation . . . . .	221
6.17.2.1	ActiveMQBytesMessageMarshaller . . . . .	221
6.17.2.2	~ActiveMQBytesMessageMarshaller . . . . .	221
6.17.3	Member Function Documentation . . . . .	221
6.17.3.1	createObject . . . . .	221
6.17.3.2	getDataStructureType . . . . .	222
6.17.3.3	looseMarshal . . . . .	222
6.17.3.4	looseUnmarshal . . . . .	222
6.17.3.5	tightMarshal1 . . . . .	223
6.17.3.6	tightMarshal2 . . . . .	223
6.17.3.7	tightUnmarshal . . . . .	224
6.18	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	
	Class Reference . . . . .	224
6.18.1	Detailed Description . . . . .	225
6.18.2	Constructor & Destructor Documentation . . . . .	225
6.18.2.1	ActiveMQBytesMessageMarshaller . . . . .	225
6.18.2.2	~ActiveMQBytesMessageMarshaller . . . . .	225
6.18.3	Member Function Documentation . . . . .	225
6.18.3.1	createObject . . . . .	225
6.18.3.2	getDataStructureType . . . . .	226



6.18.3.3	looseMarshal . . . . .	226
6.18.3.4	looseUnmarshal . . . . .	226
6.18.3.5	tightMarshal1 . . . . .	227
6.18.3.6	tightMarshal2 . . . . .	227
6.18.3.7	tightUnmarshal . . . . .	228
6.19	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller Class Reference . . . . .	228
6.19.1	Detailed Description . . . . .	229
6.19.2	Constructor & Destructor Documentation . . . . .	229
6.19.2.1	ActiveMQBytesMessageMarshaller . . . . .	229
6.19.2.2	~ActiveMQBytesMessageMarshaller . . . . .	229
6.19.3	Member Function Documentation . . . . .	229
6.19.3.1	createObject . . . . .	229
6.19.3.2	getDataStructureType . . . . .	230
6.19.3.3	looseMarshal . . . . .	230
6.19.3.4	looseUnmarshal . . . . .	230
6.19.3.5	tightMarshal1 . . . . .	231
6.19.3.6	tightMarshal2 . . . . .	231
6.19.3.7	tightUnmarshal . . . . .	232
6.20	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller Class Reference . . . . .	232
6.20.1	Detailed Description . . . . .	233
6.20.2	Constructor & Destructor Documentation . . . . .	233
6.20.2.1	ActiveMQBytesMessageMarshaller . . . . .	233
6.20.2.2	~ActiveMQBytesMessageMarshaller . . . . .	233
6.20.3	Member Function Documentation . . . . .	233
6.20.3.1	createObject . . . . .	233
6.20.3.2	getDataStructureType . . . . .	234
6.20.3.3	looseMarshal . . . . .	234
6.20.3.4	looseUnmarshal . . . . .	234
6.20.3.5	tightMarshal1 . . . . .	235
6.20.3.6	tightMarshal2 . . . . .	235
6.20.3.7	tightUnmarshal . . . . .	236
6.21	activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller Class Reference . . . . .	236

6.21.1	Detailed Description . . . . .	237
6.21.2	Constructor & Destructor Documentation . . . . .	237
6.21.2.1	ActiveMQBytesMessageMarshaller . . . . .	237
6.21.2.2	~ActiveMQBytesMessageMarshaller . . . . .	237
6.21.3	Member Function Documentation . . . . .	237
6.21.3.1	createObject . . . . .	237
6.21.3.2	getDataStructureType . . . . .	238
6.21.3.3	looseMarshal . . . . .	238
6.21.3.4	looseUnmarshal . . . . .	238
6.21.3.5	tightMarshal1 . . . . .	239
6.21.3.6	tightMarshal2 . . . . .	239
6.21.3.7	tightUnmarshal . . . . .	240
6.22	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller Class Reference . . . . .	240
6.22.1	Detailed Description . . . . .	241
6.22.2	Constructor & Destructor Documentation . . . . .	241
6.22.2.1	ActiveMQBytesMessageMarshaller . . . . .	241
6.22.2.2	~ActiveMQBytesMessageMarshaller . . . . .	241
6.22.3	Member Function Documentation . . . . .	241
6.22.3.1	createObject . . . . .	241
6.22.3.2	getDataStructureType . . . . .	242
6.22.3.3	looseMarshal . . . . .	242
6.22.3.4	looseUnmarshal . . . . .	242
6.22.3.5	tightMarshal1 . . . . .	243
6.22.3.6	tightMarshal2 . . . . .	243
6.22.3.7	tightUnmarshal . . . . .	244
6.23	activemq::core::ActiveMQConnection Class Reference . . . . .	244
6.23.1	Detailed Description . . . . .	249
6.23.2	Constructor & Destructor Documentation . . . . .	249
6.23.2.1	ActiveMQConnection . . . . .	249
6.23.2.2	~ActiveMQConnection . . . . .	249
6.23.3	Member Function Documentation . . . . .	249
6.23.3.1	addDispatcher . . . . .	249
6.23.3.2	addProducer . . . . .	250

6.23.3.3	addTransportListener . . . . .	250
6.23.3.4	close . . . . .	250
6.23.3.5	createSession . . . . .	250
6.23.3.6	createSession . . . . .	251
6.23.3.7	destroyDestination . . . . .	251
6.23.3.8	destroyDestination . . . . .	252
6.23.3.9	fire . . . . .	252
6.23.3.10	getBrokerURL . . . . .	252
6.23.3.11	getClientID . . . . .	252
6.23.3.12	getCloseTimeout . . . . .	253
6.23.3.13	getConnectionId . . . . .	253
6.23.3.14	getConnectionInfo . . . . .	253
6.23.3.15	getExceptionListener . . . . .	253
6.23.3.16	getMetaData . . . . .	254
6.23.3.17	getNextLocalTransactionId . . . . .	254
6.23.3.18	getNextSessionId . . . . .	254
6.23.3.19	getNextTempDestinationId . . . . .	254
6.23.3.20	getPassword . . . . .	255
6.23.3.21	getPrefetchPolicy . . . . .	255
6.23.3.22	getProducerWindowSize . . . . .	255
6.23.3.23	getRedeliveryPolicy . . . . .	255
6.23.3.24	getSendTimeout . . . . .	255
6.23.3.25	getTransport . . . . .	256
6.23.3.26	getUsername . . . . .	256
6.23.3.27	isAlwaysSyncSend . . . . .	256
6.23.3.28	isClosed . . . . .	256
6.23.3.29	isDispatchAsync . . . . .	256
6.23.3.30	isStarted . . . . .	256
6.23.3.31	isTransportFailed . . . . .	257
6.23.3.32	isUseAsyncSend . . . . .	257
6.23.3.33	isUseCompression . . . . .	257
6.23.3.34	onCommand . . . . .	257
6.23.3.35	oneway . . . . .	257
6.23.3.36	onException . . . . .	258

6.23.3.37	removeDispatcher . . . . .	258
6.23.3.38	removeProducer . . . . .	258
6.23.3.39	removeSession . . . . .	258
6.23.3.40	removeTransportListener . . . . .	259
6.23.3.41	sendPullRequest . . . . .	259
6.23.3.42	setAlwaysSyncSend . . . . .	259
6.23.3.43	setBrokerURL . . . . .	259
6.23.3.44	setClientID . . . . .	260
6.23.3.45	setCloseTimeout . . . . .	260
6.23.3.46	setDefaultClientID . . . . .	260
6.23.3.47	setDispatchAsync . . . . .	261
6.23.3.48	setExceptionListener . . . . .	261
6.23.3.49	setPassword . . . . .	261
6.23.3.50	setPrefetchPolicy . . . . .	261
6.23.3.51	setProducerWindowSize . . . . .	262
6.23.3.52	setRedeliveryPolicy . . . . .	262
6.23.3.53	setSendTimeout . . . . .	262
6.23.3.54	setTransportInterruptionProcessingComplete . . . . .	262
6.23.3.55	setUseAsyncSend . . . . .	262
6.23.3.56	setUseCompression . . . . .	263
6.23.3.57	setUsername . . . . .	263
6.23.3.58	start . . . . .	263
6.23.3.59	stop . . . . .	263
6.23.3.60	syncRequest . . . . .	263
6.23.3.61	transportInterrupted . . . . .	264
6.23.3.62	transportResumed . . . . .	264
6.24	activemq::core::ActiveMQConnectionFactory Class Reference . . . . .	264
6.24.1	Constructor & Destructor Documentation . . . . .	266
6.24.1.1	ActiveMQConnectionFactory . . . . .	266
6.24.1.2	ActiveMQConnectionFactory . . . . .	267
6.24.1.3	~ActiveMQConnectionFactory . . . . .	267
6.24.2	Member Function Documentation . . . . .	267
6.24.2.1	createConnection . . . . .	267
6.24.2.2	createConnection . . . . .	267

6.24.2.3	createConnection . . . . .	268
6.24.2.4	createConnection . . . . .	268
6.24.2.5	getBrokerURL . . . . .	269
6.24.2.6	getClientId . . . . .	269
6.24.2.7	getCloseTimeout . . . . .	269
6.24.2.8	getExceptionListener . . . . .	269
6.24.2.9	getPassword . . . . .	270
6.24.2.10	getPrefetchPolicy . . . . .	270
6.24.2.11	getProducerWindowSize . . . . .	270
6.24.2.12	getRedeliveryPolicy . . . . .	270
6.24.2.13	getSendTimeout . . . . .	271
6.24.2.14	getUsername . . . . .	271
6.24.2.15	isAlwaysSyncSend . . . . .	271
6.24.2.16	isDispatchAsync . . . . .	271
6.24.2.17	isUseAsyncSend . . . . .	271
6.24.2.18	isUseCompression . . . . .	271
6.24.2.19	setAlwaysSyncSend . . . . .	272
6.24.2.20	setBrokerURL . . . . .	272
6.24.2.21	setClientId . . . . .	272
6.24.2.22	setCloseTimeout . . . . .	272
6.24.2.23	setDispatchAsync . . . . .	272
6.24.2.24	setExceptionListener . . . . .	273
6.24.2.25	setPassword . . . . .	273
6.24.2.26	setPrefetchPolicy . . . . .	273
6.24.2.27	setProducerWindowSize . . . . .	273
6.24.2.28	setRedeliveryPolicy . . . . .	274
6.24.2.29	setSendTimeout . . . . .	274
6.24.2.30	setUseAsyncSend . . . . .	274
6.24.2.31	setUseCompression . . . . .	274
6.24.2.32	setUsername . . . . .	274
6.24.3	Field Documentation . . . . .	275
6.24.3.1	DEFAULT_URI . . . . .	275
6.25	activemq::core::ActiveMQConnectionMetaData Class Reference . . . .	275
6.25.1	Detailed Description . . . . .	276

6.25.2	Constructor & Destructor Documentation . . . . .	276
6.25.2.1	ActiveMQConnectionMetaData . . . . .	276
6.25.2.2	~ActiveMQConnectionMetaData . . . . .	276
6.25.3	Member Function Documentation . . . . .	276
6.25.3.1	getCMSMajorVersion . . . . .	276
6.25.3.2	getCMSMinorVersion . . . . .	276
6.25.3.3	getCMSProviderName . . . . .	277
6.25.3.4	getCMSVersion . . . . .	277
6.25.3.5	getCMSXPropertyNames . . . . .	277
6.25.3.6	getProviderMajorVersion . . . . .	278
6.25.3.7	getProviderMinorVersion . . . . .	278
6.25.3.8	getProviderVersion . . . . .	278
6.26	activemq::core::ActiveMQConstants Class Reference . . . . .	279
6.26.1	Detailed Description . . . . .	280
6.26.2	Member Enumeration Documentation . . . . .	280
6.26.2.1	AckType . . . . .	280
6.26.2.2	DestinationActions . . . . .	280
6.26.2.3	DestinationOption . . . . .	280
6.26.2.4	TransactionState . . . . .	281
6.26.2.5	URIParam . . . . .	281
6.26.3	Member Function Documentation . . . . .	282
6.26.3.1	toDestinationOption . . . . .	282
6.26.3.2	toString . . . . .	282
6.26.3.3	toString . . . . .	282
6.26.3.4	toURIOption . . . . .	282
6.27	activemq::core::ActiveMQConsumer Class Reference . . . . .	282
6.27.1	Constructor & Destructor Documentation . . . . .	284
6.27.1.1	ActiveMQConsumer . . . . .	284
6.27.1.2	~ActiveMQConsumer . . . . .	284
6.27.2	Member Function Documentation . . . . .	284
6.27.2.1	acknowledge . . . . .	285
6.27.2.2	acknowledge . . . . .	285
6.27.2.3	afterMessageIsConsumed . . . . .	285
6.27.2.4	beforeMessageIsConsumed . . . . .	285

6.27.2.5	clearMessagesInProgress . . . . .	286
6.27.2.6	close . . . . .	286
6.27.2.7	commit . . . . .	286
6.27.2.8	deliverAcks . . . . .	286
6.27.2.9	dequeue . . . . .	286
6.27.2.10	dispatch . . . . .	287
6.27.2.11	doClose . . . . .	287
6.27.2.12	getConsumerId . . . . .	287
6.27.2.13	getConsumerInfo . . . . .	287
6.27.2.14	getLastDeliveredSequenceId . . . . .	287
6.27.2.15	getMessageAvailableCount . . . . .	288
6.27.2.16	getMessageListener . . . . .	288
6.27.2.17	getMessageSelector . . . . .	288
6.27.2.18	getRedeliveryPolicy . . . . .	288
6.27.2.19	inProgressClearRequired . . . . .	289
6.27.2.20	isClosed . . . . .	289
6.27.2.21	isSynchronizationRegistered . . . . .	289
6.27.2.22	iterate . . . . .	289
6.27.2.23	receive . . . . .	289
6.27.2.24	receive . . . . .	289
6.27.2.25	receiveNoWait . . . . .	290
6.27.2.26	rollback . . . . .	290
6.27.2.27	setLastDeliveredSequenceId . . . . .	290
6.27.2.28	setMessageListener . . . . .	291
6.27.2.29	setRedeliveryPolicy . . . . .	291
6.27.2.30	setSynchronizationRegistered . . . . .	291
6.27.2.31	start . . . . .	291
6.27.2.32	stop . . . . .	291
6.28	activemq::library::ActiveMQCPP Class Reference . . . . .	292
6.28.1	Constructor & Destructor Documentation . . . . .	292
6.28.1.1	ActiveMQCPP . . . . .	292
6.28.1.2	ActiveMQCPP . . . . .	292
6.28.1.3	~ActiveMQCPP . . . . .	292
6.28.2	Member Function Documentation . . . . .	292

6.28.2.1	initializeLibrary . . . . .	292
6.28.2.2	initializeLibrary . . . . .	293
6.28.2.3	operator= . . . . .	293
6.28.2.4	shutdownLibrary . . . . .	293
6.29	activemq::commands::ActiveMQDestination Class Reference . . . . .	293
6.29.1	Constructor & Destructor Documentation . . . . .	296
6.29.1.1	ActiveMQDestination . . . . .	296
6.29.1.2	ActiveMQDestination . . . . .	296
6.29.1.3	~ActiveMQDestination . . . . .	296
6.29.2	Member Function Documentation . . . . .	296
6.29.2.1	cloneDataStructure . . . . .	296
6.29.2.2	copyDataStructure . . . . .	296
6.29.2.3	createDestination . . . . .	297
6.29.2.4	createTemporaryName . . . . .	297
6.29.2.5	equals . . . . .	297
6.29.2.6	getClientId . . . . .	298
6.29.2.7	getCMSDestination . . . . .	298
6.29.2.8	getDataStructureType . . . . .	298
6.29.2.9	getDestinationType . . . . .	298
6.29.2.10	getOptions . . . . .	299
6.29.2.11	getOrderedTarget . . . . .	299
6.29.2.12	getPhysicalName . . . . .	299
6.29.2.13	getPhysicalName . . . . .	299
6.29.2.14	isAdvisory . . . . .	299
6.29.2.15	isComposite . . . . .	299
6.29.2.16	isConnectionAdvisory . . . . .	300
6.29.2.17	isConsumerAdvisory . . . . .	300
6.29.2.18	isExclusive . . . . .	300
6.29.2.19	isOrdered . . . . .	300
6.29.2.20	isProducerAdvisory . . . . .	300
6.29.2.21	isQueue . . . . .	300
6.29.2.22	isTemporary . . . . .	301
6.29.2.23	isTopic . . . . .	301
6.29.2.24	isWildcard . . . . .	301



6.29.2.25	setAdvisory . . . . .	301
6.29.2.26	setExclusive . . . . .	301
6.29.2.27	setOrdered . . . . .	302
6.29.2.28	setOrderedTarget . . . . .	302
6.29.2.29	setPhysicalName . . . . .	302
6.29.2.30	toString . . . . .	302
6.29.3	Field Documentation . . . . .	302
6.29.3.1	advisory . . . . .	302
6.29.3.2	ADVISORY_PREFIX . . . . .	303
6.29.3.3	COMPOSITE_SEPARATOR . . . . .	303
6.29.3.4	CONNECTION_ADVISORY_PREFIX . . . . .	303
6.29.3.5	CONSUMER_ADVISORY_PREFIX . . . . .	303
6.29.3.6	DEFAULT_ORDERED_TARGET . . . . .	303
6.29.3.7	exclusive . . . . .	303
6.29.3.8	ID_ACTIVEMQDESTINATION . . . . .	303
6.29.3.9	options . . . . .	303
6.29.3.10	ordered . . . . .	303
6.29.3.11	orderedTarget . . . . .	303
6.29.3.12	physicalName . . . . .	303
6.29.3.13	PRODUCER_ADVISORY_PREFIX . . . . .	303
6.29.3.14	QUEUE_QUALIFIED_PREFIX . . . . .	304
6.29.3.15	TEMP_POSTFIX . . . . .	304
6.29.3.16	TEMP_PREFIX . . . . .	304
6.29.3.17	TEMP_QUEUE_QUALIFIED_PREFIX . . . . .	304
6.29.3.18	TEMP_TOPIC_QUALIFIED_PREFIX . . . . .	304
6.29.3.19	TOPIC_QUALIFIED_PREFIX . . . . .	304
6.30	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller Class Reference . . . . .	304
6.30.1	Detailed Description . . . . .	305
6.30.2	Constructor & Destructor Documentation . . . . .	305
6.30.2.1	ActiveMQDestinationMarshaller . . . . .	305
6.30.2.2	~ActiveMQDestinationMarshaller . . . . .	305
6.30.3	Member Function Documentation . . . . .	305
6.30.3.1	looseMarshal . . . . .	305

6.30.3.2	looseUnmarshal . . . . .	306
6.30.3.3	tightMarshal1 . . . . .	306
6.30.3.4	tightMarshal2 . . . . .	307
6.30.3.5	tightUnmarshal . . . . .	307
6.31	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller Class Reference . . . . .	308
6.31.1	Detailed Description . . . . .	309
6.31.2	Constructor & Destructor Documentation . . . . .	309
6.31.2.1	ActiveMQDestinationMarshaller . . . . .	309
6.31.2.2	~ActiveMQDestinationMarshaller . . . . .	309
6.31.3	Member Function Documentation . . . . .	309
6.31.3.1	looseMarshal . . . . .	309
6.31.3.2	looseUnmarshal . . . . .	310
6.31.3.3	tightMarshal1 . . . . .	310
6.31.3.4	tightMarshal2 . . . . .	311
6.31.3.5	tightUnmarshal . . . . .	311
6.32	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller Class Reference . . . . .	312
6.32.1	Detailed Description . . . . .	313
6.32.2	Constructor & Destructor Documentation . . . . .	313
6.32.2.1	ActiveMQDestinationMarshaller . . . . .	313
6.32.2.2	~ActiveMQDestinationMarshaller . . . . .	313
6.32.3	Member Function Documentation . . . . .	313
6.32.3.1	looseMarshal . . . . .	313
6.32.3.2	looseUnmarshal . . . . .	314
6.32.3.3	tightMarshal1 . . . . .	314
6.32.3.4	tightMarshal2 . . . . .	315
6.32.3.5	tightUnmarshal . . . . .	315
6.33	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller Class Reference . . . . .	316
6.33.1	Detailed Description . . . . .	317
6.33.2	Constructor & Destructor Documentation . . . . .	317
6.33.2.1	ActiveMQDestinationMarshaller . . . . .	317
6.33.2.2	~ActiveMQDestinationMarshaller . . . . .	317
6.33.3	Member Function Documentation . . . . .	317

6.33.3.1	looseMarshal . . . . .	317
6.33.3.2	looseUnmarshal . . . . .	318
6.33.3.3	tightMarshal1 . . . . .	318
6.33.3.4	tightMarshal2 . . . . .	319
6.33.3.5	tightUnmarshal . . . . .	319
6.34	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller Class Reference . . . . .	320
6.34.1	Detailed Description . . . . .	321
6.34.2	Constructor & Destructor Documentation . . . . .	321
6.34.2.1	ActiveMQDestinationMarshaller . . . . .	321
6.34.2.2	~ActiveMQDestinationMarshaller . . . . .	321
6.34.3	Member Function Documentation . . . . .	321
6.34.3.1	looseMarshal . . . . .	321
6.34.3.2	looseUnmarshal . . . . .	322
6.34.3.3	tightMarshal1 . . . . .	322
6.34.3.4	tightMarshal2 . . . . .	323
6.34.3.5	tightUnmarshal . . . . .	323
6.35	activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller Class Reference . . . . .	324
6.35.1	Detailed Description . . . . .	325
6.35.2	Constructor & Destructor Documentation . . . . .	325
6.35.2.1	ActiveMQDestinationMarshaller . . . . .	325
6.35.2.2	~ActiveMQDestinationMarshaller . . . . .	325
6.35.3	Member Function Documentation . . . . .	325
6.35.3.1	looseMarshal . . . . .	325
6.35.3.2	looseUnmarshal . . . . .	326
6.35.3.3	tightMarshal1 . . . . .	326
6.35.3.4	tightMarshal2 . . . . .	327
6.35.3.5	tightUnmarshal . . . . .	327
6.36	activemq::exceptions::ActiveMQException Class Reference . . . . .	328
6.36.1	Constructor & Destructor Documentation . . . . .	329
6.36.1.1	ActiveMQException . . . . .	329
6.36.1.2	ActiveMQException . . . . .	329
6.36.1.3	ActiveMQException . . . . .	329

6.36.1.4	ActiveMQException . . . . .	329
6.36.1.5	~ActiveMQException . . . . .	329
6.36.2	Member Function Documentation . . . . .	329
6.36.2.1	clone . . . . .	330
6.36.2.2	convertToCMSException . . . . .	330
6.37	activemq::commands::ActiveMQMapMessage Class Reference . . . . .	330
6.37.1	Constructor & Destructor Documentation . . . . .	333
6.37.1.1	ActiveMQMapMessage . . . . .	333
6.37.1.2	~ActiveMQMapMessage . . . . .	333
6.37.2	Member Function Documentation . . . . .	333
6.37.2.1	beforeMarshal . . . . .	333
6.37.2.2	checkMapsUnmarshalled . . . . .	333
6.37.2.3	clearBody . . . . .	333
6.37.2.4	clone . . . . .	334
6.37.2.5	cloneDataStructure . . . . .	334
6.37.2.6	copyDataStructure . . . . .	334
6.37.2.7	equals . . . . .	334
6.37.2.8	getBoolean . . . . .	335
6.37.2.9	getByte . . . . .	335
6.37.2.10	getBytes . . . . .	335
6.37.2.11	getChar . . . . .	336
6.37.2.12	getDataStructureType . . . . .	336
6.37.2.13	getDouble . . . . .	336
6.37.2.14	getFloat . . . . .	337
6.37.2.15	getInt . . . . .	337
6.37.2.16	getLong . . . . .	337
6.37.2.17	getMap . . . . .	338
6.37.2.18	getMap . . . . .	338
6.37.2.19	getMapNames . . . . .	338
6.37.2.20	getShort . . . . .	338
6.37.2.21	getString . . . . .	339
6.37.2.22	isMarshalAware . . . . .	339
6.37.2.23	itemExists . . . . .	339
6.37.2.24	setBoolean . . . . .	340

6.37.2.25	setByte . . . . .	340
6.37.2.26	setBytes . . . . .	341
6.37.2.27	setChar . . . . .	341
6.37.2.28	setDouble . . . . .	341
6.37.2.29	setFloat . . . . .	342
6.37.2.30	setInt . . . . .	342
6.37.2.31	setLong . . . . .	343
6.37.2.32	setShort . . . . .	343
6.37.2.33	setString . . . . .	343
6.37.2.34	toString . . . . .	344
6.37.3	Field Documentation . . . . .	344
6.37.3.1	ID_ACTIVEMQMAPMESSAGE . . . . .	344
6.38	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller Class Reference . . . . .	344
6.38.1	Detailed Description . . . . .	345
6.38.2	Constructor & Destructor Documentation . . . . .	345
6.38.2.1	ActiveMQMapMessageMarshaller . . . . .	345
6.38.2.2	~ActiveMQMapMessageMarshaller . . . . .	345
6.38.3	Member Function Documentation . . . . .	345
6.38.3.1	createObject . . . . .	345
6.38.3.2	getDataStructureType . . . . .	346
6.38.3.3	looseMarshal . . . . .	346
6.38.3.4	looseUnmarshal . . . . .	346
6.38.3.5	tightMarshal1 . . . . .	347
6.38.3.6	tightMarshal2 . . . . .	347
6.38.3.7	tightUnmarshal . . . . .	348
6.39	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller Class Reference . . . . .	348
6.39.1	Detailed Description . . . . .	349
6.39.2	Constructor & Destructor Documentation . . . . .	349
6.39.2.1	ActiveMQMapMessageMarshaller . . . . .	349
6.39.2.2	~ActiveMQMapMessageMarshaller . . . . .	349
6.39.3	Member Function Documentation . . . . .	349
6.39.3.1	createObject . . . . .	349

6.39.3.2	getDataStructureType . . . . .	350
6.39.3.3	looseMarshal . . . . .	350
6.39.3.4	looseUnmarshal . . . . .	350
6.39.3.5	tightMarshal1 . . . . .	351
6.39.3.6	tightMarshal2 . . . . .	351
6.39.3.7	tightUnmarshal . . . . .	352
6.40	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller Class Reference . . . . .	352
6.40.1	Detailed Description . . . . .	353
6.40.2	Constructor & Destructor Documentation . . . . .	353
6.40.2.1	ActiveMQMapMessageMarshaller . . . . .	353
6.40.2.2	~ActiveMQMapMessageMarshaller . . . . .	353
6.40.3	Member Function Documentation . . . . .	353
6.40.3.1	createObject . . . . .	353
6.40.3.2	getDataStructureType . . . . .	354
6.40.3.3	looseMarshal . . . . .	354
6.40.3.4	looseUnmarshal . . . . .	354
6.40.3.5	tightMarshal1 . . . . .	355
6.40.3.6	tightMarshal2 . . . . .	355
6.40.3.7	tightUnmarshal . . . . .	356
6.41	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller Class Reference . . . . .	356
6.41.1	Detailed Description . . . . .	357
6.41.2	Constructor & Destructor Documentation . . . . .	357
6.41.2.1	ActiveMQMapMessageMarshaller . . . . .	357
6.41.2.2	~ActiveMQMapMessageMarshaller . . . . .	357
6.41.3	Member Function Documentation . . . . .	357
6.41.3.1	createObject . . . . .	357
6.41.3.2	getDataStructureType . . . . .	358
6.41.3.3	looseMarshal . . . . .	358
6.41.3.4	looseUnmarshal . . . . .	358
6.41.3.5	tightMarshal1 . . . . .	359
6.41.3.6	tightMarshal2 . . . . .	359
6.41.3.7	tightUnmarshal . . . . .	360

6.42	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	360
6.42.1	Detailed Description . . . . .	361
6.42.2	Constructor & Destructor Documentation . . . . .	361
6.42.2.1	ActiveMQMapMessageMarshaller . . . . .	361
6.42.2.2	~ActiveMQMapMessageMarshaller . . . . .	361
6.42.3	Member Function Documentation . . . . .	361
6.42.3.1	createObject . . . . .	361
6.42.3.2	getDataStructureType . . . . .	362
6.42.3.3	looseMarshal . . . . .	362
6.42.3.4	looseUnmarshal . . . . .	362
6.42.3.5	tightMarshal1 . . . . .	363
6.42.3.6	tightMarshal2 . . . . .	363
6.42.3.7	tightUnmarshal . . . . .	364
6.43	activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	
	Class Reference . . . . .	364
6.43.1	Detailed Description . . . . .	365
6.43.2	Constructor & Destructor Documentation . . . . .	365
6.43.2.1	ActiveMQMapMessageMarshaller . . . . .	365
6.43.2.2	~ActiveMQMapMessageMarshaller . . . . .	365
6.43.3	Member Function Documentation . . . . .	365
6.43.3.1	createObject . . . . .	365
6.43.3.2	getDataStructureType . . . . .	366
6.43.3.3	looseMarshal . . . . .	366
6.43.3.4	looseUnmarshal . . . . .	366
6.43.3.5	tightMarshal1 . . . . .	367
6.43.3.6	tightMarshal2 . . . . .	367
6.43.3.7	tightUnmarshal . . . . .	368
6.44	activemq::commands::ActiveMQMessage Class Reference . . . . .	368
6.44.1	Constructor & Destructor Documentation . . . . .	369
6.44.1.1	ActiveMQMessage . . . . .	369
6.44.1.2	~ActiveMQMessage . . . . .	369
6.44.2	Member Function Documentation . . . . .	369
6.44.2.1	clone . . . . .	369

6.44.2.2	cloneDataStructure . . . . .	369
6.44.2.3	copyDataStructure . . . . .	370
6.44.2.4	equals . . . . .	370
6.44.2.5	getDataStructureType . . . . .	370
6.44.2.6	toString . . . . .	370
6.44.3	Field Documentation . . . . .	371
6.44.3.1	ID_ACTIVEMQMESSAGE . . . . .	371
6.45	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class Reference . . . . .	371
6.45.1	Detailed Description . . . . .	372
6.45.2	Constructor & Destructor Documentation . . . . .	372
6.45.2.1	ActiveMQMessageMarshaller . . . . .	372
6.45.2.2	~ActiveMQMessageMarshaller . . . . .	372
6.45.3	Member Function Documentation . . . . .	372
6.45.3.1	createObject . . . . .	372
6.45.3.2	getDataStructureType . . . . .	372
6.45.3.3	looseMarshal . . . . .	373
6.45.3.4	looseUnmarshal . . . . .	373
6.45.3.5	tightMarshal1 . . . . .	373
6.45.3.6	tightMarshal2 . . . . .	374
6.45.3.7	tightUnmarshal . . . . .	374
6.46	activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller Class Reference . . . . .	375
6.46.1	Detailed Description . . . . .	376
6.46.2	Constructor & Destructor Documentation . . . . .	376
6.46.2.1	ActiveMQMessageMarshaller . . . . .	376
6.46.2.2	~ActiveMQMessageMarshaller . . . . .	376
6.46.3	Member Function Documentation . . . . .	376
6.46.3.1	createObject . . . . .	376
6.46.3.2	getDataStructureType . . . . .	376
6.46.3.3	looseMarshal . . . . .	377
6.46.3.4	looseUnmarshal . . . . .	377
6.46.3.5	tightMarshal1 . . . . .	377
6.46.3.6	tightMarshal2 . . . . .	378



6.46.3.7	tightUnmarshal . . . . .	378
6.47	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller Class Reference . . . . .	379
6.47.1	Detailed Description . . . . .	380
6.47.2	Constructor & Destructor Documentation . . . . .	380
6.47.2.1	ActiveMQMessageMarshaller . . . . .	380
6.47.2.2	~ActiveMQMessageMarshaller . . . . .	380
6.47.3	Member Function Documentation . . . . .	380
6.47.3.1	createObject . . . . .	380
6.47.3.2	getDataStructureType . . . . .	380
6.47.3.3	looseMarshal . . . . .	381
6.47.3.4	looseUnmarshal . . . . .	381
6.47.3.5	tightMarshal1 . . . . .	381
6.47.3.6	tightMarshal2 . . . . .	382
6.47.3.7	tightUnmarshal . . . . .	382
6.48	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller Class Reference . . . . .	383
6.48.1	Detailed Description . . . . .	384
6.48.2	Constructor & Destructor Documentation . . . . .	384
6.48.2.1	ActiveMQMessageMarshaller . . . . .	384
6.48.2.2	~ActiveMQMessageMarshaller . . . . .	384
6.48.3	Member Function Documentation . . . . .	384
6.48.3.1	createObject . . . . .	384
6.48.3.2	getDataStructureType . . . . .	384
6.48.3.3	looseMarshal . . . . .	385
6.48.3.4	looseUnmarshal . . . . .	385
6.48.3.5	tightMarshal1 . . . . .	385
6.48.3.6	tightMarshal2 . . . . .	386
6.48.3.7	tightUnmarshal . . . . .	386
6.49	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller Class Reference . . . . .	387
6.49.1	Detailed Description . . . . .	388
6.49.2	Constructor & Destructor Documentation . . . . .	388
6.49.2.1	ActiveMQMessageMarshaller . . . . .	388
6.49.2.2	~ActiveMQMessageMarshaller . . . . .	388

6.49.3	Member Function Documentation . . . . .	388
6.49.3.1	createObject . . . . .	388
6.49.3.2	getDataStructureType . . . . .	388
6.49.3.3	looseMarshal . . . . .	389
6.49.3.4	looseUnmarshal . . . . .	389
6.49.3.5	tightMarshal1 . . . . .	389
6.49.3.6	tightMarshal2 . . . . .	390
6.49.3.7	tightUnmarshal . . . . .	390
6.50	activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller Class Reference . . . . .	391
6.50.1	Detailed Description . . . . .	392
6.50.2	Constructor & Destructor Documentation . . . . .	392
6.50.2.1	ActiveMQMessageMarshaller . . . . .	392
6.50.2.2	~ActiveMQMessageMarshaller . . . . .	392
6.50.3	Member Function Documentation . . . . .	392
6.50.3.1	createObject . . . . .	392
6.50.3.2	getDataStructureType . . . . .	392
6.50.3.3	looseMarshal . . . . .	393
6.50.3.4	looseUnmarshal . . . . .	393
6.50.3.5	tightMarshal1 . . . . .	393
6.50.3.6	tightMarshal2 . . . . .	394
6.50.3.7	tightUnmarshal . . . . .	394
6.51	activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference . . . . .	395
6.51.1	Constructor & Destructor Documentation . . . . .	398
6.51.1.1	ActiveMQMessageTemplate . . . . .	398
6.51.1.2	~ActiveMQMessageTemplate . . . . .	398
6.51.2	Member Function Documentation . . . . .	398
6.51.2.1	acknowledge . . . . .	398
6.51.2.2	clearBody . . . . .	398
6.51.2.3	clearProperties . . . . .	399
6.51.2.4	equals . . . . .	399
6.51.2.5	failIfReadOnlyBody . . . . .	399
6.51.2.6	failIfReadOnlyProperties . . . . .	399

6.51.2.7	failIfWriteOnlyBody . . . . .	399
6.51.2.8	getBooleanProperty . . . . .	399
6.51.2.9	getByteProperty . . . . .	400
6.51.2.10	getCMSCorrelationID . . . . .	400
6.51.2.11	getCMSDeliveryMode . . . . .	401
6.51.2.12	getCMSDestination . . . . .	401
6.51.2.13	getCMSExpiration . . . . .	401
6.51.2.14	getCMSMessageID . . . . .	402
6.51.2.15	getCMSPriority . . . . .	402
6.51.2.16	getCMSRedelivered . . . . .	402
6.51.2.17	getCMSReplyTo . . . . .	403
6.51.2.18	getCMSTimestamp . . . . .	403
6.51.2.19	getCMSType . . . . .	403
6.51.2.20	getDoubleProperty . . . . .	404
6.51.2.21	getFloatProperty . . . . .	404
6.51.2.22	getIntProperty . . . . .	405
6.51.2.23	getLongProperty . . . . .	405
6.51.2.24	getPropertyNames . . . . .	406
6.51.2.25	getShortProperty . . . . .	406
6.51.2.26	getStringProperty . . . . .	406
6.51.2.27	onSend . . . . .	407
6.51.2.28	propertyExists . . . . .	407
6.51.2.29	setBooleanProperty . . . . .	407
6.51.2.30	setByteProperty . . . . .	408
6.51.2.31	setCMSCorrelationID . . . . .	408
6.51.2.32	setCMSDeliveryMode . . . . .	408
6.51.2.33	setCMSDestination . . . . .	409
6.51.2.34	setCMSExpiration . . . . .	409
6.51.2.35	setCMSMessageID . . . . .	409
6.51.2.36	setCMSPriority . . . . .	410
6.51.2.37	setCMSRedelivered . . . . .	410
6.51.2.38	setCMSReplyTo . . . . .	410
6.51.2.39	setCMSTimestamp . . . . .	411
6.51.2.40	setCMSType . . . . .	411

6.51.2.41	setDoubleProperty . . . . .	411
6.51.2.42	setFloatProperty . . . . .	412
6.51.2.43	setIntProperty . . . . .	412
6.51.2.44	setLongProperty . . . . .	412
6.51.2.45	setShortProperty . . . . .	413
6.51.2.46	setStringProperty . . . . .	413
6.52	activemq::commands::ActiveMQObjectMessage Class Reference . . . .	414
6.52.1	Constructor & Destructor Documentation . . . . .	414
6.52.1.1	ActiveMQObjectMessage . . . . .	414
6.52.1.2	~ActiveMQObjectMessage . . . . .	414
6.52.2	Member Function Documentation . . . . .	415
6.52.2.1	clone . . . . .	415
6.52.2.2	cloneDataStructure . . . . .	415
6.52.2.3	copyDataStructure . . . . .	415
6.52.2.4	equals . . . . .	415
6.52.2.5	getDataStructureType . . . . .	416
6.52.2.6	toString . . . . .	416
6.52.3	Field Documentation . . . . .	416
6.52.3.1	ID_ACTIVEMQOBJECTMESSAGE . . . . .	416
6.53	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference . . . . .	416
6.53.1	Detailed Description . . . . .	417
6.53.2	Constructor & Destructor Documentation . . . . .	417
6.53.2.1	ActiveMQObjectMessageMarshaller . . . . .	417
6.53.2.2	~ActiveMQObjectMessageMarshaller . . . . .	417
6.53.3	Member Function Documentation . . . . .	417
6.53.3.1	createObject . . . . .	418
6.53.3.2	getDataStructureType . . . . .	418
6.53.3.3	looseMarshal . . . . .	418
6.53.3.4	looseUnmarshal . . . . .	419
6.53.3.5	tightMarshal1 . . . . .	419
6.53.3.6	tightMarshal2 . . . . .	420
6.53.3.7	tightUnmarshal . . . . .	420

6.54	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	421
6.54.1	Detailed Description . . . . .	421
6.54.2	Constructor & Destructor Documentation . . . . .	422
6.54.2.1	ActiveMQObjectMessageMarshaller . . . . .	422
6.54.2.2	~ActiveMQObjectMessageMarshaller . . . . .	422
6.54.3	Member Function Documentation . . . . .	422
6.54.3.1	createObject . . . . .	422
6.54.3.2	getDataStructureType . . . . .	422
6.54.3.3	looseMarshal . . . . .	422
6.54.3.4	looseUnmarshal . . . . .	423
6.54.3.5	tightMarshal1 . . . . .	423
6.54.3.6	tightMarshal2 . . . . .	424
6.54.3.7	tightUnmarshal . . . . .	424
6.55	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	425
6.55.1	Detailed Description . . . . .	425
6.55.2	Constructor & Destructor Documentation . . . . .	426
6.55.2.1	ActiveMQObjectMessageMarshaller . . . . .	426
6.55.2.2	~ActiveMQObjectMessageMarshaller . . . . .	426
6.55.3	Member Function Documentation . . . . .	426
6.55.3.1	createObject . . . . .	426
6.55.3.2	getDataStructureType . . . . .	426
6.55.3.3	looseMarshal . . . . .	426
6.55.3.4	looseUnmarshal . . . . .	427
6.55.3.5	tightMarshal1 . . . . .	427
6.55.3.6	tightMarshal2 . . . . .	428
6.55.3.7	tightUnmarshal . . . . .	428
6.56	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	429
6.56.1	Detailed Description . . . . .	429
6.56.2	Constructor & Destructor Documentation . . . . .	430
6.56.2.1	ActiveMQObjectMessageMarshaller . . . . .	430
6.56.2.2	~ActiveMQObjectMessageMarshaller . . . . .	430
6.56.3	Member Function Documentation . . . . .	430

6.56.3.1	createObject . . . . .	430
6.56.3.2	getDataStructureType . . . . .	430
6.56.3.3	looseMarshal . . . . .	430
6.56.3.4	looseUnmarshal . . . . .	431
6.56.3.5	tightMarshal1 . . . . .	431
6.56.3.6	tightMarshal2 . . . . .	432
6.56.3.7	tightUnmarshal . . . . .	432
6.57	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	433
6.57.1	Detailed Description . . . . .	433
6.57.2	Constructor & Destructor Documentation . . . . .	434
6.57.2.1	ActiveMQObjectMessageMarshaller . . . . .	434
6.57.2.2	~ActiveMQObjectMessageMarshaller . . . . .	434
6.57.3	Member Function Documentation . . . . .	434
6.57.3.1	createObject . . . . .	434
6.57.3.2	getDataStructureType . . . . .	434
6.57.3.3	looseMarshal . . . . .	434
6.57.3.4	looseUnmarshal . . . . .	435
6.57.3.5	tightMarshal1 . . . . .	435
6.57.3.6	tightMarshal2 . . . . .	436
6.57.3.7	tightUnmarshal . . . . .	436
6.58	activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	
	Class Reference . . . . .	437
6.58.1	Detailed Description . . . . .	437
6.58.2	Constructor & Destructor Documentation . . . . .	438
6.58.2.1	ActiveMQObjectMessageMarshaller . . . . .	438
6.58.2.2	~ActiveMQObjectMessageMarshaller . . . . .	438
6.58.3	Member Function Documentation . . . . .	438
6.58.3.1	createObject . . . . .	438
6.58.3.2	getDataStructureType . . . . .	438
6.58.3.3	looseMarshal . . . . .	438
6.58.3.4	looseUnmarshal . . . . .	439
6.58.3.5	tightMarshal1 . . . . .	439
6.58.3.6	tightMarshal2 . . . . .	440

6.58.3.7	tightUnmarshal . . . . .	440
6.59	activemq::core::ActiveMQProducer Class Reference . . . . .	441
6.59.1	Constructor & Destructor Documentation . . . . .	442
6.59.1.1	ActiveMQProducer . . . . .	442
6.59.1.2	~ActiveMQProducer . . . . .	443
6.59.2	Member Function Documentation . . . . .	443
6.59.2.1	close . . . . .	443
6.59.2.2	getDeliveryMode . . . . .	443
6.59.2.3	getDisableMessageID . . . . .	443
6.59.2.4	getDisableMessageTimeStamp . . . . .	443
6.59.2.5	getPriority . . . . .	444
6.59.2.6	getProducerId . . . . .	444
6.59.2.7	getProducerInfo . . . . .	444
6.59.2.8	getSendTimeout . . . . .	444
6.59.2.9	getTimeToLive . . . . .	444
6.59.2.10	isClosed . . . . .	445
6.59.2.11	onProducerAck . . . . .	445
6.59.2.12	send . . . . .	445
6.59.2.13	send . . . . .	446
6.59.2.14	send . . . . .	446
6.59.2.15	send . . . . .	447
6.59.2.16	setDeliveryMode . . . . .	447
6.59.2.17	setDisableMessageID . . . . .	447
6.59.2.18	setDisableMessageTimeStamp . . . . .	448
6.59.2.19	setPriority . . . . .	448
6.59.2.20	setSendTimeout . . . . .	448
6.59.2.21	setTimeToLive . . . . .	448
6.60	activemq::util::ActiveMQProperties Class Reference . . . . .	449
6.60.1	Detailed Description . . . . .	450
6.60.2	Constructor & Destructor Documentation . . . . .	450
6.60.2.1	ActiveMQProperties . . . . .	450
6.60.2.2	~ActiveMQProperties . . . . .	450
6.60.3	Member Function Documentation . . . . .	450
6.60.3.1	clear . . . . .	450

6.60.3.2	clone . . . . .	450
6.60.3.3	copy . . . . .	450
6.60.3.4	getProperties . . . . .	450
6.60.3.5	getProperties . . . . .	451
6.60.3.6	getProperty . . . . .	451
6.60.3.7	getProperty . . . . .	451
6.60.3.8	hasProperty . . . . .	451
6.60.3.9	isEmpty . . . . .	452
6.60.3.10	remove . . . . .	452
6.60.3.11	setProperties . . . . .	452
6.60.3.12	setProperty . . . . .	452
6.60.3.13	toArray . . . . .	452
6.60.3.14	toString . . . . .	453
6.61	activemq::commands::ActiveMQQueue Class Reference . . . . .	453
6.61.1	Constructor & Destructor Documentation . . . . .	454
6.61.1.1	ActiveMQQueue . . . . .	454
6.61.1.2	ActiveMQQueue . . . . .	454
6.61.1.3	~ActiveMQQueue . . . . .	454
6.61.2	Member Function Documentation . . . . .	454
6.61.2.1	clone . . . . .	454
6.61.2.2	cloneDataStructure . . . . .	454
6.61.2.3	copy . . . . .	455
6.61.2.4	copyDataStructure . . . . .	455
6.61.2.5	equals . . . . .	455
6.61.2.6	getCMSDestination . . . . .	455
6.61.2.7	getCMSProperties . . . . .	456
6.61.2.8	getDataStructureType . . . . .	456
6.61.2.9	getDestinationType . . . . .	456
6.61.2.10	getQueueName . . . . .	456
6.61.2.11	toString . . . . .	457
6.61.3	Field Documentation . . . . .	457
6.61.3.1	ID_ACTIVEMQQUEUE . . . . .	457
6.62	activemq::core::ActiveMQQueueBrowser Class Reference . . . . .	457
6.62.1	Constructor & Destructor Documentation . . . . .	458



6.62.1.1	ActiveMQQueueBrowser . . . . .	458
6.62.1.2	~ActiveMQQueueBrowser . . . . .	458
6.62.2	Member Function Documentation . . . . .	458
6.62.2.1	close . . . . .	458
6.62.2.2	getEnumeration . . . . .	458
6.62.2.3	getMessageSelector . . . . .	459
6.62.2.4	getQueue . . . . .	459
6.62.2.5	hasMoreMessages . . . . .	459
6.62.2.6	nextMessage . . . . .	460
6.62.3	Friends And Related Function Documentation . . . . .	460
6.62.3.1	Browser . . . . .	460
6.63	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference . . . . .	460
6.63.1	Detailed Description . . . . .	461
6.63.2	Constructor & Destructor Documentation . . . . .	461
6.63.2.1	ActiveMQQueueMarshaller . . . . .	461
6.63.2.2	~ActiveMQQueueMarshaller . . . . .	461
6.63.3	Member Function Documentation . . . . .	461
6.63.3.1	createObject . . . . .	461
6.63.3.2	getDataStructureType . . . . .	462
6.63.3.3	looseMarshal . . . . .	462
6.63.3.4	looseUnmarshal . . . . .	462
6.63.3.5	tightMarshal1 . . . . .	463
6.63.3.6	tightMarshal2 . . . . .	463
6.63.3.7	tightUnmarshal . . . . .	464
6.64	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class Reference . . . . .	464
6.64.1	Detailed Description . . . . .	465
6.64.2	Constructor & Destructor Documentation . . . . .	465
6.64.2.1	ActiveMQQueueMarshaller . . . . .	465
6.64.2.2	~ActiveMQQueueMarshaller . . . . .	465
6.64.3	Member Function Documentation . . . . .	465
6.64.3.1	createObject . . . . .	465
6.64.3.2	getDataStructureType . . . . .	466

6.64.3.3	looseMarshal . . . . .	466
6.64.3.4	looseUnmarshal . . . . .	466
6.64.3.5	tightMarshal1 . . . . .	467
6.64.3.6	tightMarshal2 . . . . .	467
6.64.3.7	tightUnmarshal . . . . .	468
6.65	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	
	Class Reference . . . . .	468
6.65.1	Detailed Description . . . . .	469
6.65.2	Constructor & Destructor Documentation . . . . .	469
6.65.2.1	ActiveMQQueueMarshaller . . . . .	469
6.65.2.2	~ActiveMQQueueMarshaller . . . . .	469
6.65.3	Member Function Documentation . . . . .	469
6.65.3.1	createObject . . . . .	469
6.65.3.2	getDataStructureType . . . . .	470
6.65.3.3	looseMarshal . . . . .	470
6.65.3.4	looseUnmarshal . . . . .	470
6.65.3.5	tightMarshal1 . . . . .	471
6.65.3.6	tightMarshal2 . . . . .	471
6.65.3.7	tightUnmarshal . . . . .	472
6.66	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	
	Class Reference . . . . .	472
6.66.1	Detailed Description . . . . .	473
6.66.2	Constructor & Destructor Documentation . . . . .	473
6.66.2.1	ActiveMQQueueMarshaller . . . . .	473
6.66.2.2	~ActiveMQQueueMarshaller . . . . .	473
6.66.3	Member Function Documentation . . . . .	473
6.66.3.1	createObject . . . . .	473
6.66.3.2	getDataStructureType . . . . .	474
6.66.3.3	looseMarshal . . . . .	474
6.66.3.4	looseUnmarshal . . . . .	474
6.66.3.5	tightMarshal1 . . . . .	475
6.66.3.6	tightMarshal2 . . . . .	475
6.66.3.7	tightUnmarshal . . . . .	476
6.67	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	
	Class Reference . . . . .	476

6.67.1	Detailed Description . . . . .	477
6.67.2	Constructor & Destructor Documentation . . . . .	477
6.67.2.1	ActiveMQQueueMarshaller . . . . .	477
6.67.2.2	~ActiveMQQueueMarshaller . . . . .	477
6.67.3	Member Function Documentation . . . . .	477
6.67.3.1	createObject . . . . .	477
6.67.3.2	getDataStructureType . . . . .	478
6.67.3.3	looseMarshal . . . . .	478
6.67.3.4	looseUnmarshal . . . . .	478
6.67.3.5	tightMarshal1 . . . . .	479
6.67.3.6	tightMarshal2 . . . . .	479
6.67.3.7	tightUnmarshal . . . . .	480
6.68	activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller Class Reference . . . . .	480
6.68.1	Detailed Description . . . . .	481
6.68.2	Constructor & Destructor Documentation . . . . .	481
6.68.2.1	ActiveMQQueueMarshaller . . . . .	481
6.68.2.2	~ActiveMQQueueMarshaller . . . . .	481
6.68.3	Member Function Documentation . . . . .	481
6.68.3.1	createObject . . . . .	481
6.68.3.2	getDataStructureType . . . . .	482
6.68.3.3	looseMarshal . . . . .	482
6.68.3.4	looseUnmarshal . . . . .	482
6.68.3.5	tightMarshal1 . . . . .	483
6.68.3.6	tightMarshal2 . . . . .	483
6.68.3.7	tightUnmarshal . . . . .	484
6.69	activemq::core::ActiveMQSession Class Reference . . . . .	484
6.69.1	Constructor & Destructor Documentation . . . . .	488
6.69.1.1	ActiveMQSession . . . . .	488
6.69.1.2	~ActiveMQSession . . . . .	488
6.69.2	Member Function Documentation . . . . .	488
6.69.2.1	acknowledge . . . . .	488
6.69.2.2	addConsumer . . . . .	488
6.69.2.3	addProducer . . . . .	489

6.69.2.4	clearMessagesInProgress . . . . .	489
6.69.2.5	close . . . . .	489
6.69.2.6	commit . . . . .	489
6.69.2.7	createBrowser . . . . .	489
6.69.2.8	createBrowser . . . . .	490
6.69.2.9	createBytesMessage . . . . .	490
6.69.2.10	createBytesMessage . . . . .	491
6.69.2.11	createConsumer . . . . .	491
6.69.2.12	createConsumer . . . . .	492
6.69.2.13	createConsumer . . . . .	492
6.69.2.14	createDurableConsumer . . . . .	492
6.69.2.15	createMapMessage . . . . .	493
6.69.2.16	createMessage . . . . .	493
6.69.2.17	createProducer . . . . .	493
6.69.2.18	createQueue . . . . .	494
6.69.2.19	createStreamMessage . . . . .	494
6.69.2.20	createTemporaryQueue . . . . .	494
6.69.2.21	createTemporaryTopic . . . . .	494
6.69.2.22	createTextMessage . . . . .	495
6.69.2.23	createTextMessage . . . . .	495
6.69.2.24	createTopic . . . . .	495
6.69.2.25	deliverAcks . . . . .	496
6.69.2.26	dispatch . . . . .	496
6.69.2.27	doStartTransaction . . . . .	496
6.69.2.28	fire . . . . .	496
6.69.2.29	getAcknowledgeMode . . . . .	496
6.69.2.30	getConnection . . . . .	497
6.69.2.31	getExceptionListener . . . . .	497
6.69.2.32	getLastDeliveredSequenceId . . . . .	497
6.69.2.33	getNextConsumerId . . . . .	497
6.69.2.34	getNextProducerId . . . . .	497
6.69.2.35	getSessionId . . . . .	498
6.69.2.36	getSessionInfo . . . . .	498
6.69.2.37	getTransactionContext . . . . .	498

6.69.2.38	isAutoAcknowledge . . . . .	498
6.69.2.39	isClientAcknowledge . . . . .	498
6.69.2.40	isDupsOkAcknowledge . . . . .	498
6.69.2.41	isIndividualAcknowledge . . . . .	499
6.69.2.42	isStarted . . . . .	499
6.69.2.43	isTransacted . . . . .	499
6.69.2.44	oneway . . . . .	499
6.69.2.45	recover . . . . .	499
6.69.2.46	redispatch . . . . .	500
6.69.2.47	removeConsumer . . . . .	500
6.69.2.48	removeProducer . . . . .	500
6.69.2.49	rollback . . . . .	501
6.69.2.50	send . . . . .	501
6.69.2.51	setLastDeliveredSequenceId . . . . .	501
6.69.2.52	start . . . . .	502
6.69.2.53	stop . . . . .	502
6.69.2.54	syncRequest . . . . .	502
6.69.2.55	unsubscribe . . . . .	502
6.69.2.56	wakeup . . . . .	503
6.69.3	Friends And Related Function Documentation . . . . .	503
6.69.3.1	ActiveMQSessionExecutor . . . . .	503
6.70	activemq::core::ActiveMQSessionExecutor Class Reference . . . . .	503
6.70.1	Detailed Description . . . . .	504
6.70.2	Constructor & Destructor Documentation . . . . .	504
6.70.2.1	ActiveMQSessionExecutor . . . . .	504
6.70.2.2	~ActiveMQSessionExecutor . . . . .	504
6.70.3	Member Function Documentation . . . . .	504
6.70.3.1	clear . . . . .	504
6.70.3.2	clearMessagesInProgress . . . . .	504
6.70.3.3	close . . . . .	504
6.70.3.4	execute . . . . .	505
6.70.3.5	executeFirst . . . . .	505
6.70.3.6	getUnconsumedMessages . . . . .	505
6.70.3.7	hasUnconsumedMessages . . . . .	505

6.70.3.8	isEmpty	505
6.70.3.9	isRunning	506
6.70.3.10	iterate	506
6.70.3.11	start	506
6.70.3.12	stop	506
6.70.3.13	wakeup	506
6.71	activemq::commands::ActiveMQStreamMessage Class Reference	506
6.71.1	Constructor & Destructor Documentation	509
6.71.1.1	ActiveMQStreamMessage	509
6.71.1.2	~ActiveMQStreamMessage	509
6.71.2	Member Function Documentation	509
6.71.2.1	clearBody	509
6.71.2.2	clone	509
6.71.2.3	cloneDataStructure	510
6.71.2.4	copyDataStructure	510
6.71.2.5	equals	510
6.71.2.6	getDataStructureType	510
6.71.2.7	onSend	511
6.71.2.8	readBoolean	511
6.71.2.9	readByte	511
6.71.2.10	readBytes	512
6.71.2.11	readBytes	513
6.71.2.12	readChar	513
6.71.2.13	readDouble	514
6.71.2.14	readFloat	514
6.71.2.15	readInt	515
6.71.2.16	readLong	515
6.71.2.17	readShort	516
6.71.2.18	readString	516
6.71.2.19	readUnsignedShort	517
6.71.2.20	reset	517
6.71.2.21	toString	518
6.71.2.22	writeBoolean	518
6.71.2.23	writeByte	518

6.71.2.24	writeBytes	519
6.71.2.25	writeBytes	519
6.71.2.26	writeChar	520
6.71.2.27	writeDouble	520
6.71.2.28	writeFloat	520
6.71.2.29	writeInt	521
6.71.2.30	writeLong	521
6.71.2.31	writeShort	522
6.71.2.32	writeString	522
6.71.2.33	writeUnsignedShort	522
6.71.3	Field Documentation	523
6.71.3.1	ID_ACTIVEMQSTREAMMESSAGE	523
6.72	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	
	Class Reference	523
6.72.1	Detailed Description	524
6.72.2	Constructor & Destructor Documentation	524
6.72.2.1	ActiveMQStreamMessageMarshaller	524
6.72.2.2	~ActiveMQStreamMessageMarshaller	524
6.72.3	Member Function Documentation	524
6.72.3.1	createObject	524
6.72.3.2	getDataStructureType	524
6.72.3.3	looseMarshal	525
6.72.3.4	looseUnmarshal	525
6.72.3.5	tightMarshal1	526
6.72.3.6	tightMarshal2	526
6.72.3.7	tightUnmarshal	527
6.73	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	
	Class Reference	527
6.73.1	Detailed Description	528
6.73.2	Constructor & Destructor Documentation	528
6.73.2.1	ActiveMQStreamMessageMarshaller	528
6.73.2.2	~ActiveMQStreamMessageMarshaller	528
6.73.3	Member Function Documentation	528
6.73.3.1	createObject	528

6.73.3.2	getDataStructureType . . . . .	529
6.73.3.3	looseMarshal . . . . .	529
6.73.3.4	looseUnmarshal . . . . .	529
6.73.3.5	tightMarshal1 . . . . .	530
6.73.3.6	tightMarshal2 . . . . .	530
6.73.3.7	tightUnmarshal . . . . .	531
6.74	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller Class Reference . . . . .	531
6.74.1	Detailed Description . . . . .	532
6.74.2	Constructor & Destructor Documentation . . . . .	532
6.74.2.1	ActiveMQStreamMessageMarshaller . . . . .	532
6.74.2.2	~ActiveMQStreamMessageMarshaller . . . . .	532
6.74.3	Member Function Documentation . . . . .	532
6.74.3.1	createObject . . . . .	532
6.74.3.2	getDataStructureType . . . . .	533
6.74.3.3	looseMarshal . . . . .	533
6.74.3.4	looseUnmarshal . . . . .	533
6.74.3.5	tightMarshal1 . . . . .	534
6.74.3.6	tightMarshal2 . . . . .	534
6.74.3.7	tightUnmarshal . . . . .	535
6.75	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller Class Reference . . . . .	535
6.75.1	Detailed Description . . . . .	536
6.75.2	Constructor & Destructor Documentation . . . . .	536
6.75.2.1	ActiveMQStreamMessageMarshaller . . . . .	536
6.75.2.2	~ActiveMQStreamMessageMarshaller . . . . .	536
6.75.3	Member Function Documentation . . . . .	536
6.75.3.1	createObject . . . . .	536
6.75.3.2	getDataStructureType . . . . .	537
6.75.3.3	looseMarshal . . . . .	537
6.75.3.4	looseUnmarshal . . . . .	537
6.75.3.5	tightMarshal1 . . . . .	538
6.75.3.6	tightMarshal2 . . . . .	538
6.75.3.7	tightUnmarshal . . . . .	539



6.76	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	539
6.76.1	Detailed Description . . . . .	540
6.76.2	Constructor & Destructor Documentation . . . . .	540
6.76.2.1	ActiveMQStreamMessageMarshaller . . . . .	540
6.76.2.2	~ActiveMQStreamMessageMarshaller . . . . .	540
6.76.3	Member Function Documentation . . . . .	540
6.76.3.1	createObject . . . . .	540
6.76.3.2	getDataStructureType . . . . .	541
6.76.3.3	looseMarshal . . . . .	541
6.76.3.4	looseUnmarshal . . . . .	541
6.76.3.5	tightMarshal1 . . . . .	542
6.76.3.6	tightMarshal2 . . . . .	542
6.76.3.7	tightUnmarshal . . . . .	543
6.77	activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	
	Class Reference . . . . .	543
6.77.1	Detailed Description . . . . .	544
6.77.2	Constructor & Destructor Documentation . . . . .	544
6.77.2.1	ActiveMQStreamMessageMarshaller . . . . .	544
6.77.2.2	~ActiveMQStreamMessageMarshaller . . . . .	544
6.77.3	Member Function Documentation . . . . .	544
6.77.3.1	createObject . . . . .	544
6.77.3.2	getDataStructureType . . . . .	545
6.77.3.3	looseMarshal . . . . .	545
6.77.3.4	looseUnmarshal . . . . .	545
6.77.3.5	tightMarshal1 . . . . .	546
6.77.3.6	tightMarshal2 . . . . .	546
6.77.3.7	tightUnmarshal . . . . .	547
6.78	activemq::commands::ActiveMQTempDestination Class Reference . . . . .	547
6.78.1	Constructor & Destructor Documentation . . . . .	548
6.78.1.1	ActiveMQTempDestination . . . . .	548
6.78.1.2	ActiveMQTempDestination . . . . .	548
6.78.1.3	~ActiveMQTempDestination . . . . .	548
6.78.2	Member Function Documentation . . . . .	548

6.78.2.1	cloneDataStructure . . . . .	548
6.78.2.2	close . . . . .	549
6.78.2.3	copyDataStructure . . . . .	549
6.78.2.4	equals . . . . .	549
6.78.2.5	getDataStructureType . . . . .	550
6.78.2.6	setConnection . . . . .	550
6.78.2.7	toString . . . . .	550
6.78.3	Field Documentation . . . . .	550
6.78.3.1	connection . . . . .	550
6.78.3.2	ID_ACTIVEMQTEMPDESTINATION . . . . .	551
6.79	activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller Class Reference . . . . .	551
6.79.1	Detailed Description . . . . .	552
6.79.2	Constructor & Destructor Documentation . . . . .	552
6.79.2.1	ActiveMQTempDestinationMarshaller . . . . .	552
6.79.2.2	~ActiveMQTempDestinationMarshaller . . . . .	552
6.79.3	Member Function Documentation . . . . .	552
6.79.3.1	looseMarshal . . . . .	552
6.79.3.2	looseUnmarshal . . . . .	552
6.79.3.3	tightMarshal1 . . . . .	553
6.79.3.4	tightMarshal2 . . . . .	554
6.79.3.5	tightUnmarshal . . . . .	554
6.80	activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller Class Reference . . . . .	555
6.80.1	Detailed Description . . . . .	555
6.80.2	Constructor & Destructor Documentation . . . . .	556
6.80.2.1	ActiveMQTempDestinationMarshaller . . . . .	556
6.80.2.2	~ActiveMQTempDestinationMarshaller . . . . .	556
6.80.3	Member Function Documentation . . . . .	556
6.80.3.1	looseMarshal . . . . .	556
6.80.3.2	looseUnmarshal . . . . .	556
6.80.3.3	tightMarshal1 . . . . .	557
6.80.3.4	tightMarshal2 . . . . .	557
6.80.3.5	tightUnmarshal . . . . .	558

6.81	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	558
6.81.1	Detailed Description . . . . .	559
6.81.2	Constructor & Destructor Documentation . . . . .	559
6.81.2.1	ActiveMQTempDestinationMarshaller . . . . .	559
6.81.2.2	~ActiveMQTempDestinationMarshaller . . . . .	559
6.81.3	Member Function Documentation . . . . .	560
6.81.3.1	looseMarshal . . . . .	560
6.81.3.2	looseUnmarshal . . . . .	560
6.81.3.3	tightMarshal1 . . . . .	561
6.81.3.4	tightMarshal2 . . . . .	561
6.81.3.5	tightUnmarshal . . . . .	562
6.82	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	562
6.82.1	Detailed Description . . . . .	563
6.82.2	Constructor & Destructor Documentation . . . . .	563
6.82.2.1	ActiveMQTempDestinationMarshaller . . . . .	563
6.82.2.2	~ActiveMQTempDestinationMarshaller . . . . .	563
6.82.3	Member Function Documentation . . . . .	563
6.82.3.1	looseMarshal . . . . .	563
6.82.3.2	looseUnmarshal . . . . .	564
6.82.3.3	tightMarshal1 . . . . .	564
6.82.3.4	tightMarshal2 . . . . .	565
6.82.3.5	tightUnmarshal . . . . .	566
6.83	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller	
	Class Reference . . . . .	566
6.83.1	Detailed Description . . . . .	567
6.83.2	Constructor & Destructor Documentation . . . . .	567
6.83.2.1	ActiveMQTempDestinationMarshaller . . . . .	567
6.83.2.2	~ActiveMQTempDestinationMarshaller . . . . .	567
6.83.3	Member Function Documentation . . . . .	567
6.83.3.1	looseMarshal . . . . .	567
6.83.3.2	looseUnmarshal . . . . .	568
6.83.3.3	tightMarshal1 . . . . .	568
6.83.3.4	tightMarshal2 . . . . .	569

6.83.3.5	tightUnmarshal . . . . .	569
6.84	activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller Class Reference . . . . .	570
6.84.1	Detailed Description . . . . .	571
6.84.2	Constructor & Destructor Documentation . . . . .	571
6.84.2.1	ActiveMQTempDestinationMarshaller . . . . .	571
6.84.2.2	~ActiveMQTempDestinationMarshaller . . . . .	571
6.84.3	Member Function Documentation . . . . .	571
6.84.3.1	looseMarshal . . . . .	571
6.84.3.2	looseUnmarshal . . . . .	572
6.84.3.3	tightMarshal1 . . . . .	572
6.84.3.4	tightMarshal2 . . . . .	573
6.84.3.5	tightUnmarshal . . . . .	573
6.85	activemq::commands::ActiveMQTempQueue Class Reference . . . . .	574
6.85.1	Constructor & Destructor Documentation . . . . .	575
6.85.1.1	ActiveMQTempQueue . . . . .	575
6.85.1.2	ActiveMQTempQueue . . . . .	575
6.85.1.3	~ActiveMQTempQueue . . . . .	575
6.85.2	Member Function Documentation . . . . .	575
6.85.2.1	clone . . . . .	575
6.85.2.2	cloneDataStructure . . . . .	575
6.85.2.3	copy . . . . .	576
6.85.2.4	copyDataStructure . . . . .	576
6.85.2.5	destroy . . . . .	576
6.85.2.6	equals . . . . .	576
6.85.2.7	getCMSDestination . . . . .	577
6.85.2.8	getCMSProperties . . . . .	577
6.85.2.9	getDataStructureType . . . . .	577
6.85.2.10	getDestinationType . . . . .	577
6.85.2.11	getQueueName . . . . .	578
6.85.2.12	toString . . . . .	578
6.85.3	Field Documentation . . . . .	578
6.85.3.1	ID_ACTIVEMQTEMPQUEUE . . . . .	578

6.86	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	578
6.86.1	Detailed Description . . . . .	579
6.86.2	Constructor & Destructor Documentation . . . . .	579
6.86.2.1	ActiveMQTempQueueMarshaller . . . . .	579
6.86.2.2	~ActiveMQTempQueueMarshaller . . . . .	579
6.86.3	Member Function Documentation . . . . .	579
6.86.3.1	createObject . . . . .	579
6.86.3.2	getDataStructureType . . . . .	580
6.86.3.3	looseMarshal . . . . .	580
6.86.3.4	looseUnmarshal . . . . .	580
6.86.3.5	tightMarshal1 . . . . .	581
6.86.3.6	tightMarshal2 . . . . .	581
6.86.3.7	tightUnmarshal . . . . .	582
6.87	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	582
6.87.1	Detailed Description . . . . .	583
6.87.2	Constructor & Destructor Documentation . . . . .	583
6.87.2.1	ActiveMQTempQueueMarshaller . . . . .	583
6.87.2.2	~ActiveMQTempQueueMarshaller . . . . .	583
6.87.3	Member Function Documentation . . . . .	583
6.87.3.1	createObject . . . . .	583
6.87.3.2	getDataStructureType . . . . .	584
6.87.3.3	looseMarshal . . . . .	584
6.87.3.4	looseUnmarshal . . . . .	584
6.87.3.5	tightMarshal1 . . . . .	585
6.87.3.6	tightMarshal2 . . . . .	585
6.87.3.7	tightUnmarshal . . . . .	586
6.88	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	
	Class Reference . . . . .	586
6.88.1	Detailed Description . . . . .	587
6.88.2	Constructor & Destructor Documentation . . . . .	587
6.88.2.1	ActiveMQTempQueueMarshaller . . . . .	587
6.88.2.2	~ActiveMQTempQueueMarshaller . . . . .	587
6.88.3	Member Function Documentation . . . . .	587

6.88.3.1	createObject . . . . .	587
6.88.3.2	getDataStructureType . . . . .	588
6.88.3.3	looseMarshal . . . . .	588
6.88.3.4	looseUnmarshal . . . . .	588
6.88.3.5	tightMarshal1 . . . . .	589
6.88.3.6	tightMarshal2 . . . . .	589
6.88.3.7	tightUnmarshal . . . . .	590
6.89	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller Class Reference . . . . .	590
6.89.1	Detailed Description . . . . .	591
6.89.2	Constructor & Destructor Documentation . . . . .	591
6.89.2.1	ActiveMQTempQueueMarshaller . . . . .	591
6.89.2.2	~ActiveMQTempQueueMarshaller . . . . .	591
6.89.3	Member Function Documentation . . . . .	591
6.89.3.1	createObject . . . . .	591
6.89.3.2	getDataStructureType . . . . .	592
6.89.3.3	looseMarshal . . . . .	592
6.89.3.4	looseUnmarshal . . . . .	592
6.89.3.5	tightMarshal1 . . . . .	593
6.89.3.6	tightMarshal2 . . . . .	593
6.89.3.7	tightUnmarshal . . . . .	594
6.90	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference . . . . .	594
6.90.1	Detailed Description . . . . .	595
6.90.2	Constructor & Destructor Documentation . . . . .	595
6.90.2.1	ActiveMQTempQueueMarshaller . . . . .	595
6.90.2.2	~ActiveMQTempQueueMarshaller . . . . .	595
6.90.3	Member Function Documentation . . . . .	595
6.90.3.1	createObject . . . . .	595
6.90.3.2	getDataStructureType . . . . .	596
6.90.3.3	looseMarshal . . . . .	596
6.90.3.4	looseUnmarshal . . . . .	596
6.90.3.5	tightMarshal1 . . . . .	597
6.90.3.6	tightMarshal2 . . . . .	597

6.90.3.7	tightUnmarshal . . . . .	598
6.91	activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller Class Reference . . . . .	598
6.91.1	Detailed Description . . . . .	599
6.91.2	Constructor & Destructor Documentation . . . . .	599
6.91.2.1	ActiveMQTempQueueMarshaller . . . . .	599
6.91.2.2	~ActiveMQTempQueueMarshaller . . . . .	599
6.91.3	Member Function Documentation . . . . .	599
6.91.3.1	createObject . . . . .	599
6.91.3.2	getDataStructureType . . . . .	600
6.91.3.3	looseMarshal . . . . .	600
6.91.3.4	looseUnmarshal . . . . .	600
6.91.3.5	tightMarshal1 . . . . .	601
6.91.3.6	tightMarshal2 . . . . .	601
6.91.3.7	tightUnmarshal . . . . .	602
6.92	activemq::commands::ActiveMQTempTopic Class Reference . . . . .	602
6.92.1	Constructor & Destructor Documentation . . . . .	603
6.92.1.1	ActiveMQTempTopic . . . . .	603
6.92.1.2	ActiveMQTempTopic . . . . .	603
6.92.1.3	~ActiveMQTempTopic . . . . .	603
6.92.2	Member Function Documentation . . . . .	603
6.92.2.1	clone . . . . .	603
6.92.2.2	cloneDataStructure . . . . .	604
6.92.2.3	copy . . . . .	604
6.92.2.4	copyDataStructure . . . . .	604
6.92.2.5	destroy . . . . .	604
6.92.2.6	equals . . . . .	605
6.92.2.7	getCMSDestination . . . . .	605
6.92.2.8	getCMSProperties . . . . .	605
6.92.2.9	getDataStructureType . . . . .	605
6.92.2.10	getDestinationType . . . . .	606
6.92.2.11	getTopicName . . . . .	606
6.92.2.12	toString . . . . .	606
6.92.3	Field Documentation . . . . .	606

6.92.3.1	ID_ACTIVEMQTEMPTOPIC . . . . .	606
6.93	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller Class Reference . . . . .	607
6.93.1	Detailed Description . . . . .	607
6.93.2	Constructor & Destructor Documentation . . . . .	608
6.93.2.1	ActiveMQTempTopicMarshaller . . . . .	608
6.93.2.2	~ActiveMQTempTopicMarshaller . . . . .	608
6.93.3	Member Function Documentation . . . . .	608
6.93.3.1	createObject . . . . .	608
6.93.3.2	getDataStructureType . . . . .	608
6.93.3.3	looseMarshal . . . . .	608
6.93.3.4	looseUnmarshal . . . . .	609
6.93.3.5	tightMarshal1 . . . . .	609
6.93.3.6	tightMarshal2 . . . . .	610
6.93.3.7	tightUnmarshal . . . . .	610
6.94	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller Class Reference . . . . .	611
6.94.1	Detailed Description . . . . .	611
6.94.2	Constructor & Destructor Documentation . . . . .	612
6.94.2.1	ActiveMQTempTopicMarshaller . . . . .	612
6.94.2.2	~ActiveMQTempTopicMarshaller . . . . .	612
6.94.3	Member Function Documentation . . . . .	612
6.94.3.1	createObject . . . . .	612
6.94.3.2	getDataStructureType . . . . .	612
6.94.3.3	looseMarshal . . . . .	612
6.94.3.4	looseUnmarshal . . . . .	613
6.94.3.5	tightMarshal1 . . . . .	613
6.94.3.6	tightMarshal2 . . . . .	614
6.94.3.7	tightUnmarshal . . . . .	614
6.95	activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller Class Reference . . . . .	615
6.95.1	Detailed Description . . . . .	615
6.95.2	Constructor & Destructor Documentation . . . . .	616
6.95.2.1	ActiveMQTempTopicMarshaller . . . . .	616
6.95.2.2	~ActiveMQTempTopicMarshaller . . . . .	616



6.95.3	Member Function Documentation . . . . .	616
6.95.3.1	createObject . . . . .	616
6.95.3.2	getDataStructureType . . . . .	616
6.95.3.3	looseMarshal . . . . .	616
6.95.3.4	looseUnmarshal . . . . .	617
6.95.3.5	tightMarshal1 . . . . .	617
6.95.3.6	tightMarshal2 . . . . .	618
6.95.3.7	tightUnmarshal . . . . .	618
6.96	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller Class Reference . . . . .	619
6.96.1	Detailed Description . . . . .	619
6.96.2	Constructor & Destructor Documentation . . . . .	620
6.96.2.1	ActiveMQTempTopicMarshaller . . . . .	620
6.96.2.2	~ActiveMQTempTopicMarshaller . . . . .	620
6.96.3	Member Function Documentation . . . . .	620
6.96.3.1	createObject . . . . .	620
6.96.3.2	getDataStructureType . . . . .	620
6.96.3.3	looseMarshal . . . . .	620
6.96.3.4	looseUnmarshal . . . . .	621
6.96.3.5	tightMarshal1 . . . . .	621
6.96.3.6	tightMarshal2 . . . . .	622
6.96.3.7	tightUnmarshal . . . . .	622
6.97	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller Class Reference . . . . .	623
6.97.1	Detailed Description . . . . .	623
6.97.2	Constructor & Destructor Documentation . . . . .	624
6.97.2.1	ActiveMQTempTopicMarshaller . . . . .	624
6.97.2.2	~ActiveMQTempTopicMarshaller . . . . .	624
6.97.3	Member Function Documentation . . . . .	624
6.97.3.1	createObject . . . . .	624
6.97.3.2	getDataStructureType . . . . .	624
6.97.3.3	looseMarshal . . . . .	624
6.97.3.4	looseUnmarshal . . . . .	625
6.97.3.5	tightMarshal1 . . . . .	625

6.97.3.6	tightMarshal2 . . . . .	626
6.97.3.7	tightUnmarshal . . . . .	626
6.98	activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller Class Reference . . . . .	627
6.98.1	Detailed Description . . . . .	627
6.98.2	Constructor & Destructor Documentation . . . . .	628
6.98.2.1	ActiveMQTempTopicMarshaller . . . . .	628
6.98.2.2	~ActiveMQTempTopicMarshaller . . . . .	628
6.98.3	Member Function Documentation . . . . .	628
6.98.3.1	createObject . . . . .	628
6.98.3.2	getDataStructureType . . . . .	628
6.98.3.3	looseMarshal . . . . .	628
6.98.3.4	looseUnmarshal . . . . .	629
6.98.3.5	tightMarshal1 . . . . .	629
6.98.3.6	tightMarshal2 . . . . .	630
6.98.3.7	tightUnmarshal . . . . .	630
6.99	activemq::commands::ActiveMQTextMessage Class Reference . . . . .	631
6.99.1	Constructor & Destructor Documentation . . . . .	632
6.99.1.1	ActiveMQTextMessage . . . . .	632
6.99.1.2	~ActiveMQTextMessage . . . . .	632
6.99.2	Member Function Documentation . . . . .	632
6.99.2.1	beforeMarshal . . . . .	632
6.99.2.2	clearBody . . . . .	632
6.99.2.3	clone . . . . .	632
6.99.2.4	cloneDataStructure . . . . .	633
6.99.2.5	copyDataStructure . . . . .	633
6.99.2.6	equals . . . . .	633
6.99.2.7	getDataStructureType . . . . .	633
6.99.2.8	getSize . . . . .	634
6.99.2.9	getText . . . . .	634
6.99.2.10	setText . . . . .	634
6.99.2.11	setText . . . . .	635
6.99.2.12	toString . . . . .	635
6.99.3	Field Documentation . . . . .	635

6.99.3.1	ID_ACTIVEMQTEXTMESSAGE . . . . .	635
6.99.3.2	text . . . . .	635
6.100	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference . . . . .	635
6.100.1	Detailed Description . . . . .	636
6.100.2	Constructor & Destructor Documentation . . . . .	636
6.100.2.1	ActiveMQTextMessageMarshaller . . . . .	636
6.100.2.2	~ActiveMQTextMessageMarshaller . . . . .	637
6.100.3	Member Function Documentation . . . . .	637
6.100.3.1	createObject . . . . .	637
6.100.3.2	getDataStructureType . . . . .	637
6.100.3.3	looseMarshal . . . . .	637
6.100.3.4	looseUnmarshal . . . . .	638
6.100.3.5	tightMarshal1 . . . . .	638
6.100.3.6	tightMarshal2 . . . . .	639
6.100.3.7	tightUnmarshal . . . . .	639
6.101	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller Class Reference . . . . .	640
6.101.1	Detailed Description . . . . .	640
6.101.2	Constructor & Destructor Documentation . . . . .	641
6.101.2.1	ActiveMQTextMessageMarshaller . . . . .	641
6.101.2.2	~ActiveMQTextMessageMarshaller . . . . .	641
6.101.3	Member Function Documentation . . . . .	641
6.101.3.1	createObject . . . . .	641
6.101.3.2	getDataStructureType . . . . .	641
6.101.3.3	looseMarshal . . . . .	641
6.101.3.4	looseUnmarshal . . . . .	642
6.101.3.5	tightMarshal1 . . . . .	642
6.101.3.6	tightMarshal2 . . . . .	643
6.101.3.7	tightUnmarshal . . . . .	643
6.102	activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller Class Reference . . . . .	644
6.102.1	Detailed Description . . . . .	644
6.102.2	Constructor & Destructor Documentation . . . . .	645
6.102.2.1	ActiveMQTextMessageMarshaller . . . . .	645

6.102.2.2 ~ActiveMQTextMessageMarshaller . . . . .	645
6.102.3 Member Function Documentation . . . . .	645
6.102.3.1 createObject . . . . .	645
6.102.3.2 getDataStructureType . . . . .	645
6.102.3.3 looseMarshal . . . . .	645
6.102.3.4 looseUnmarshal . . . . .	646
6.102.3.5 tightMarshal1 . . . . .	646
6.102.3.6 tightMarshal2 . . . . .	647
6.102.3.7 tightUnmarshal . . . . .	647
6.103activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	
Class Reference . . . . .	648
6.103.1 Detailed Description . . . . .	648
6.103.2 Constructor & Destructor Documentation . . . . .	649
6.103.2.1 ActiveMQTextMessageMarshaller . . . . .	649
6.103.2.2 ~ActiveMQTextMessageMarshaller . . . . .	649
6.103.3 Member Function Documentation . . . . .	649
6.103.3.1 createObject . . . . .	649
6.103.3.2 getDataStructureType . . . . .	649
6.103.3.3 looseMarshal . . . . .	649
6.103.3.4 looseUnmarshal . . . . .	650
6.103.3.5 tightMarshal1 . . . . .	650
6.103.3.6 tightMarshal2 . . . . .	651
6.103.3.7 tightUnmarshal . . . . .	651
6.104activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	
Class Reference . . . . .	652
6.104.1 Detailed Description . . . . .	652
6.104.2 Constructor & Destructor Documentation . . . . .	653
6.104.2.1 ActiveMQTextMessageMarshaller . . . . .	653
6.104.2.2 ~ActiveMQTextMessageMarshaller . . . . .	653
6.104.3 Member Function Documentation . . . . .	653
6.104.3.1 createObject . . . . .	653
6.104.3.2 getDataStructureType . . . . .	653
6.104.3.3 looseMarshal . . . . .	653
6.104.3.4 looseUnmarshal . . . . .	654

6.104.3.5	tightMarshal1 . . . . .	654
6.104.3.6	tightMarshal2 . . . . .	655
6.104.3.7	tightUnmarshal . . . . .	655
6.105	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	
	Class Reference . . . . .	656
6.105.1	Detailed Description . . . . .	656
6.105.2	Constructor & Destructor Documentation . . . . .	657
6.105.2.1	ActiveMQTextMessageMarshaller . . . . .	657
6.105.2.2	~ActiveMQTextMessageMarshaller . . . . .	657
6.105.3	Member Function Documentation . . . . .	657
6.105.3.1	createObject . . . . .	657
6.105.3.2	getDataStructureType . . . . .	657
6.105.3.3	looseMarshal . . . . .	657
6.105.3.4	looseUnmarshal . . . . .	658
6.105.3.5	tightMarshal1 . . . . .	658
6.105.3.6	tightMarshal2 . . . . .	659
6.105.3.7	tightUnmarshal . . . . .	659
6.106	activemq::commands::ActiveMQTopic Class Reference . . . . .	660
6.106.1	Constructor & Destructor Documentation . . . . .	661
6.106.1.1	ActiveMQTopic . . . . .	661
6.106.1.2	ActiveMQTopic . . . . .	661
6.106.1.3	~ActiveMQTopic . . . . .	661
6.106.2	Member Function Documentation . . . . .	661
6.106.2.1	clone . . . . .	661
6.106.2.2	cloneDataStructure . . . . .	661
6.106.2.3	copy . . . . .	661
6.106.2.4	copyDataStructure . . . . .	662
6.106.2.5	equals . . . . .	662
6.106.2.6	getCMSDestination . . . . .	662
6.106.2.7	getCMSProperties . . . . .	662
6.106.2.8	getDataStructureType . . . . .	663
6.106.2.9	getDestinationType . . . . .	663
6.106.2.10	getTopicName . . . . .	663
6.106.2.11	toString . . . . .	663

6.106.3 Field Documentation . . . . .	664
6.106.3.1 ID_ACTIVEMQTOPIC . . . . .	664
6.107activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference . . . . .	664
6.107.1 Detailed Description . . . . .	665
6.107.2 Constructor & Destructor Documentation . . . . .	665
6.107.2.1 ActiveMQTopicMarshaller . . . . .	665
6.107.2.2 ~ActiveMQTopicMarshaller . . . . .	665
6.107.3 Member Function Documentation . . . . .	665
6.107.3.1 createObject . . . . .	665
6.107.3.2 getDataStructureType . . . . .	665
6.107.3.3 looseMarshal . . . . .	666
6.107.3.4 looseUnmarshal . . . . .	666
6.107.3.5 tightMarshal1 . . . . .	666
6.107.3.6 tightMarshal2 . . . . .	667
6.107.3.7 tightUnmarshal . . . . .	667
6.108activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference . . . . .	668
6.108.1 Detailed Description . . . . .	669
6.108.2 Constructor & Destructor Documentation . . . . .	669
6.108.2.1 ActiveMQTopicMarshaller . . . . .	669
6.108.2.2 ~ActiveMQTopicMarshaller . . . . .	669
6.108.3 Member Function Documentation . . . . .	669
6.108.3.1 createObject . . . . .	669
6.108.3.2 getDataStructureType . . . . .	669
6.108.3.3 looseMarshal . . . . .	670
6.108.3.4 looseUnmarshal . . . . .	670
6.108.3.5 tightMarshal1 . . . . .	670
6.108.3.6 tightMarshal2 . . . . .	671
6.108.3.7 tightUnmarshal . . . . .	671
6.109activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference . . . . .	672
6.109.1 Detailed Description . . . . .	673
6.109.2 Constructor & Destructor Documentation . . . . .	673
6.109.2.1 ActiveMQTopicMarshaller . . . . .	673

6.109.2.2	~ActiveMQTopicMarshaller . . . . .	673
6.109.3	Member Function Documentation . . . . .	673
6.109.3.1	createObject . . . . .	673
6.109.3.2	getDataStructureType . . . . .	673
6.109.3.3	looseMarshal . . . . .	674
6.109.3.4	looseUnmarshal . . . . .	674
6.109.3.5	tightMarshal1 . . . . .	674
6.109.3.6	tightMarshal2 . . . . .	675
6.109.3.7	tightUnmarshal . . . . .	675
6.110	activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference . . . . .	676
6.110.1	Detailed Description . . . . .	677
6.110.2	Constructor & Destructor Documentation . . . . .	677
6.110.2.1	ActiveMQTopicMarshaller . . . . .	677
6.110.2.2	~ActiveMQTopicMarshaller . . . . .	677
6.110.3	Member Function Documentation . . . . .	677
6.110.3.1	createObject . . . . .	677
6.110.3.2	getDataStructureType . . . . .	677
6.110.3.3	looseMarshal . . . . .	678
6.110.3.4	looseUnmarshal . . . . .	678
6.110.3.5	tightMarshal1 . . . . .	678
6.110.3.6	tightMarshal2 . . . . .	679
6.110.3.7	tightUnmarshal . . . . .	679
6.111	activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller Class Reference . . . . .	680
6.111.1	Detailed Description . . . . .	681
6.111.2	Constructor & Destructor Documentation . . . . .	681
6.111.2.1	ActiveMQTopicMarshaller . . . . .	681
6.111.2.2	~ActiveMQTopicMarshaller . . . . .	681
6.111.3	Member Function Documentation . . . . .	681
6.111.3.1	createObject . . . . .	681
6.111.3.2	getDataStructureType . . . . .	681
6.111.3.3	looseMarshal . . . . .	682
6.111.3.4	looseUnmarshal . . . . .	682

6.111.3.5 tightMarshal1 . . . . .	682
6.111.3.6 tightMarshal2 . . . . .	683
6.111.3.7 tightUnmarshal . . . . .	683
6.112activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference . . . . .	684
6.112.1 Detailed Description . . . . .	685
6.112.2 Constructor & Destructor Documentation . . . . .	685
6.112.2.1 ActiveMQTopicMarshaller . . . . .	685
6.112.2.2 ~ActiveMQTopicMarshaller . . . . .	685
6.112.3 Member Function Documentation . . . . .	685
6.112.3.1 createObject . . . . .	685
6.112.3.2 getDataStructureType . . . . .	685
6.112.3.3 looseMarshal . . . . .	686
6.112.3.4 looseUnmarshal . . . . .	686
6.112.3.5 tightMarshal1 . . . . .	686
6.112.3.6 tightMarshal2 . . . . .	687
6.112.3.7 tightUnmarshal . . . . .	687
6.113activemq::core::ActiveMQTransactionContext Class Reference . . . . .	688
6.113.1 Detailed Description . . . . .	689
6.113.2 Constructor & Destructor Documentation . . . . .	689
6.113.2.1 ActiveMQTransactionContext . . . . .	689
6.113.2.2 ~ActiveMQTransactionContext . . . . .	689
6.113.3 Member Function Documentation . . . . .	689
6.113.3.1 addSynchronization . . . . .	689
6.113.3.2 begin . . . . .	689
6.113.3.3 commit . . . . .	690
6.113.3.4 getTransactionId . . . . .	690
6.113.3.5 isInTransaction . . . . .	690
6.113.3.6 removeSynchronization . . . . .	690
6.113.3.7 rollback . . . . .	691
6.114decaf::util::zip::Adler32 Class Reference . . . . .	691
6.114.1 Detailed Description . . . . .	691
6.114.2 Constructor & Destructor Documentation . . . . .	692
6.114.2.1 Adler32 . . . . .	692



6.114.2.2 ~Adler32 . . . . .	692
6.114.3 Member Function Documentation . . . . .	692
6.114.3.1 getValue . . . . .	692
6.114.3.2 reset . . . . .	692
6.114.3.3 update . . . . .	692
6.114.3.4 update . . . . .	693
6.114.3.5 update . . . . .	693
6.114.3.6 update . . . . .	693
6.115decaf::lang::Appendable Class Reference . . . . .	693
6.115.1 Detailed Description . . . . .	694
6.115.2 Constructor & Destructor Documentation . . . . .	694
6.115.2.1 ~Appendable . . . . .	694
6.115.3 Member Function Documentation . . . . .	694
6.115.3.1 append . . . . .	694
6.115.3.2 append . . . . .	695
6.115.3.3 append . . . . .	695
6.116decaf::internal::AprPool Class Reference . . . . .	696
6.116.1 Detailed Description . . . . .	696
6.116.2 Constructor & Destructor Documentation . . . . .	696
6.116.2.1 AprPool . . . . .	696
6.116.2.2 ~AprPool . . . . .	696
6.116.3 Member Function Documentation . . . . .	697
6.116.3.1 cleanup . . . . .	697
6.116.3.2 getAprPool . . . . .	697
6.116.3.3 getGlobalPool . . . . .	697
6.117decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Refer- ence . . . . .	697
6.117.1 Detailed Description . . . . .	699
6.117.2 Member Typedef Documentation . . . . .	699
6.117.2.1 ConstReferenceType . . . . .	699
6.117.2.2 CounterType . . . . .	699
6.117.2.3 PointerType . . . . .	699
6.117.2.4 ReferenceType . . . . .	699
6.117.3 Constructor & Destructor Documentation . . . . .	699

6.117.3.1 ArrayPointer . . . . .	699
6.117.3.2 ArrayPointer . . . . .	700
6.117.3.3 ArrayPointer . . . . .	700
6.117.3.4 ArrayPointer . . . . .	700
6.117.3.5 ~ArrayPointer . . . . .	700
6.117.4 Member Function Documentation . . . . .	700
6.117.4.1 clone . . . . .	701
6.117.4.2 get . . . . .	701
6.117.4.3 length . . . . .	701
6.117.4.4 operator! . . . . .	701
6.117.4.5 operator!= . . . . .	702
6.117.4.6 operator= . . . . .	702
6.117.4.7 operator= . . . . .	702
6.117.4.8 operator== . . . . .	702
6.117.4.9 operator[] . . . . .	702
6.117.4.10operator[] . . . . .	702
6.117.4.11release . . . . .	703
6.117.4.12reset . . . . .	703
6.117.4.13swap . . . . .	703
6.117.5 Friends And Related Function Documentation . . . . .	704
6.117.5.1 operator!= . . . . .	704
6.117.5.2 operator!= . . . . .	704
6.117.5.3 operator== . . . . .	704
6.117.5.4 operator== . . . . .	704
6.118decaf::lang::ArrayPointerComparator< T, R > Class Template Reference	704
6.118.1 Detailed Description . . . . .	704
6.118.2 Member Function Documentation . . . . .	705
6.118.2.1 compare . . . . .	705
6.118.2.2 operator() . . . . .	705
6.119decaf::util::concurrent::atomic::AtomicBoolean Class Reference . . . . .	705
6.119.1 Detailed Description . . . . .	706
6.119.2 Constructor & Destructor Documentation . . . . .	706
6.119.2.1 AtomicBoolean . . . . .	706
6.119.2.2 AtomicBoolean . . . . .	706

6.119.2.3 ~AtomicBoolean . . . . .	706
6.119.3 Member Function Documentation . . . . .	706
6.119.3.1 compareAndSet . . . . .	706
6.119.3.2 get . . . . .	707
6.119.3.3 getAndSet . . . . .	707
6.119.3.4 set . . . . .	707
6.119.3.5 toString . . . . .	707
6.120decaf::util::concurrent::atomic::AtomicInteger Class Reference . . . . .	708
6.120.1 Detailed Description . . . . .	709
6.120.2 Constructor & Destructor Documentation . . . . .	709
6.120.2.1 AtomicInteger . . . . .	709
6.120.2.2 AtomicInteger . . . . .	709
6.120.2.3 ~AtomicInteger . . . . .	709
6.120.3 Member Function Documentation . . . . .	709
6.120.3.1 addAndGet . . . . .	709
6.120.3.2 compareAndSet . . . . .	710
6.120.3.3 decrementAndGet . . . . .	710
6.120.3.4 doubleValue . . . . .	710
6.120.3.5 floatValue . . . . .	711
6.120.3.6 get . . . . .	711
6.120.3.7 getAndAdd . . . . .	711
6.120.3.8 getAndDecrement . . . . .	711
6.120.3.9 getAndIncrement . . . . .	711
6.120.3.10getAndSet . . . . .	712
6.120.3.11incrementAndGet . . . . .	712
6.120.3.12intValue . . . . .	712
6.120.3.13longValue . . . . .	712
6.120.3.14set . . . . .	713
6.120.3.15toString . . . . .	713
6.121decaf::util::concurrent::atomic::AtomicRefCounter Class Reference . . . . .	713
6.121.1 Constructor & Destructor Documentation . . . . .	714
6.121.1.1 AtomicRefCounter . . . . .	714
6.121.1.2 AtomicRefCounter . . . . .	714
6.121.1.3 ~AtomicRefCounter . . . . .	714

6.121.2 Member Function Documentation . . . . .	714
6.121.2.1 release . . . . .	714
6.121.2.2 swap . . . . .	715
6.122decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference . . . . .	716
6.122.1 Detailed Description . . . . .	716
6.122.2 Constructor & Destructor Documentation . . . . .	716
6.122.2.1 AtomicReference . . . . .	716
6.122.2.2 AtomicReference . . . . .	716
6.122.2.3 ~AtomicReference . . . . .	716
6.122.3 Member Function Documentation . . . . .	717
6.122.3.1 compareAndSet . . . . .	717
6.122.3.2 get . . . . .	717
6.122.3.3 getAndSet . . . . .	717
6.122.3.4 set . . . . .	717
6.122.3.5 toString . . . . .	718
6.123activemq::transport::failover::BackupTransport Class Reference . . . . .	718
6.123.1 Constructor & Destructor Documentation . . . . .	719
6.123.1.1 BackupTransport . . . . .	719
6.123.1.2 ~BackupTransport . . . . .	719
6.123.2 Member Function Documentation . . . . .	719
6.123.2.1 getTransport . . . . .	719
6.123.2.2 getUri . . . . .	719
6.123.2.3 isClosed . . . . .	719
6.123.2.4 onException . . . . .	719
6.123.2.5 setClosed . . . . .	720
6.123.2.6 setTransport . . . . .	720
6.123.2.7 setUri . . . . .	720
6.124activemq::transport::failover::BackupTransportPool Class Reference . . . . .	720
6.124.1 Constructor & Destructor Documentation . . . . .	721
6.124.1.1 BackupTransportPool . . . . .	721
6.124.1.2 BackupTransportPool . . . . .	721
6.124.1.3 ~BackupTransportPool . . . . .	721
6.124.2 Member Function Documentation . . . . .	721

6.124.2.1	getBackup	722
6.124.2.2	getBackupPoolSize	722
6.124.2.3	isEnabled	722
6.124.2.4	isPending	722
6.124.2.5	iterate	722
6.124.2.6	setBackupPoolSize	723
6.124.2.7	setEnabled	723
6.124.3	Friends And Related Function Documentation	723
6.124.3.1	BackupTransport	723
6.125	activemq::commands::BaseCommand Class Reference	723
6.125.1	Constructor & Destructor Documentation	724
6.125.1.1	BaseCommand	724
6.125.1.2	~BaseCommand	724
6.125.2	Member Function Documentation	724
6.125.2.1	copyDataStructure	724
6.125.2.2	equals	725
6.125.2.3	getCommandId	726
6.125.2.4	isBrokerInfo	726
6.125.2.5	isConnectionInfo	726
6.125.2.6	isConsumerInfo	726
6.125.2.7	isKeepAliveInfo	727
6.125.2.8	isMessage	727
6.125.2.9	isMessageAck	727
6.125.2.10	sMessageDispatch	727
6.125.2.11	isMessageDispatchNotification	727
6.125.2.12	sProducerAck	727
6.125.2.13	sProducerInfo	727
6.125.2.14	sRemoveInfo	728
6.125.2.15	sRemoveSubscriptionInfo	728
6.125.2.16	sResponse	728
6.125.2.17	sResponseRequired	728
6.125.2.18	sShutdownInfo	728
6.125.2.19	sTransactionInfo	728
6.125.2.20	sWireFormatInfo	729

6.125.2.21	setCommandId . . . . .	729
6.125.2.22	setResponseRequired . . . . .	729
6.125.2.23	toString . . . . .	729
6.126	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	
	Class Reference . . . . .	730
6.126.1	Detailed Description . . . . .	731
6.126.2	Constructor & Destructor Documentation . . . . .	731
6.126.2.1	BaseCommandMarshaller . . . . .	731
6.126.2.2	~BaseCommandMarshaller . . . . .	731
6.126.3	Member Function Documentation . . . . .	731
6.126.3.1	looseMarshal . . . . .	731
6.126.3.2	looseUnmarshal . . . . .	732
6.126.3.3	tightMarshal1 . . . . .	733
6.126.3.4	tightMarshal2 . . . . .	734
6.126.3.5	tightUnmarshal . . . . .	736
6.127	activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	
	Class Reference . . . . .	737
6.127.1	Detailed Description . . . . .	738
6.127.2	Constructor & Destructor Documentation . . . . .	738
6.127.2.1	BaseCommandMarshaller . . . . .	738
6.127.2.2	~BaseCommandMarshaller . . . . .	738
6.127.3	Member Function Documentation . . . . .	738
6.127.3.1	looseMarshal . . . . .	738
6.127.3.2	looseUnmarshal . . . . .	739
6.127.3.3	tightMarshal1 . . . . .	740
6.127.3.4	tightMarshal2 . . . . .	741
6.127.3.5	tightUnmarshal . . . . .	742
6.128	activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	
	Class Reference . . . . .	743
6.128.1	Detailed Description . . . . .	744
6.128.2	Constructor & Destructor Documentation . . . . .	744
6.128.2.1	BaseCommandMarshaller . . . . .	744
6.128.2.2	~BaseCommandMarshaller . . . . .	744
6.128.3	Member Function Documentation . . . . .	744
6.128.3.1	looseMarshal . . . . .	745

6.128.3.2 looseUnmarshal . . . . .	746
6.128.3.3 tightMarshal1 . . . . .	747
6.128.3.4 tightMarshal2 . . . . .	748
6.128.3.5 tightUnmarshal . . . . .	749
6.129activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	
Class Reference . . . . .	750
6.129.1 Detailed Description . . . . .	751
6.129.2 Constructor & Destructor Documentation . . . . .	751
6.129.2.1 BaseCommandMarshaller . . . . .	751
6.129.2.2 ~BaseCommandMarshaller . . . . .	751
6.129.3 Member Function Documentation . . . . .	751
6.129.3.1 looseMarshal . . . . .	751
6.129.3.2 looseUnmarshal . . . . .	752
6.129.3.3 tightMarshal1 . . . . .	754
6.129.3.4 tightMarshal2 . . . . .	755
6.129.3.5 tightUnmarshal . . . . .	756
6.130activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller	
Class Reference . . . . .	757
6.130.1 Detailed Description . . . . .	758
6.130.2 Constructor & Destructor Documentation . . . . .	758
6.130.2.1 BaseCommandMarshaller . . . . .	758
6.130.2.2 ~BaseCommandMarshaller . . . . .	758
6.130.3 Member Function Documentation . . . . .	758
6.130.3.1 looseMarshal . . . . .	758
6.130.3.2 looseUnmarshal . . . . .	759
6.130.3.3 tightMarshal1 . . . . .	760
6.130.3.4 tightMarshal2 . . . . .	762
6.130.3.5 tightUnmarshal . . . . .	763
6.131activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	
Class Reference . . . . .	764
6.131.1 Detailed Description . . . . .	765
6.131.2 Constructor & Destructor Documentation . . . . .	765
6.131.2.1 BaseCommandMarshaller . . . . .	765
6.131.2.2 ~BaseCommandMarshaller . . . . .	765
6.131.3 Member Function Documentation . . . . .	765

6.131.3.1 looseMarshal . . . . .	765
6.131.3.2 looseUnmarshal . . . . .	766
6.131.3.3 tightMarshal1 . . . . .	767
6.131.3.4 tightMarshal2 . . . . .	768
6.131.3.5 tightUnmarshal . . . . .	769
6.132activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference . . . . .	770
6.132.1 Detailed Description . . . . .	776
6.132.2 Constructor & Destructor Documentation . . . . .	776
6.132.2.1 ~BaseDataStreamMarshaller . . . . .	776
6.132.3 Member Function Documentation . . . . .	776
6.132.3.1 looseMarshal . . . . .	776
6.132.3.2 looseMarshalBrokerError . . . . .	776
6.132.3.3 looseMarshalCachedObject . . . . .	777
6.132.3.4 looseMarshalLong . . . . .	777
6.132.3.5 looseMarshalNestedObject . . . . .	777
6.132.3.6 looseMarshalObjectArray . . . . .	778
6.132.3.7 looseMarshalString . . . . .	778
6.132.3.8 looseUnmarshal . . . . .	779
6.132.3.9 looseUnmarshalBrokerError . . . . .	779
6.132.3.10looseUnmarshalByteArray . . . . .	779
6.132.3.11looseUnmarshalCachedObject . . . . .	780
6.132.3.12looseUnmarshalConstByteArray . . . . .	780
6.132.3.13looseUnmarshalLong . . . . .	781
6.132.3.14looseUnmarshalNestedObject . . . . .	781
6.132.3.15looseUnmarshalString . . . . .	781
6.132.3.16readAsciiString . . . . .	782
6.132.3.17tightMarshal1 . . . . .	782
6.132.3.18tightMarshal2 . . . . .	782
6.132.3.19tightMarshalBrokerError1 . . . . .	783
6.132.3.20tightMarshalBrokerError2 . . . . .	783
6.132.3.21tightMarshalCachedObject1 . . . . .	784
6.132.3.22tightMarshalCachedObject2 . . . . .	784
6.132.3.23tightMarshalLong1 . . . . .	784



6.132.3.24	tightMarshalLong2 . . . . .	785
6.132.3.25	tightMarshalNestedObject1 . . . . .	785
6.132.3.26	tightMarshalNestedObject2 . . . . .	786
6.132.3.27	tightMarshalObjectArray1 . . . . .	786
6.132.3.28	tightMarshalObjectArray2 . . . . .	787
6.132.3.29	tightMarshalString1 . . . . .	787
6.132.3.30	tightMarshalString2 . . . . .	788
6.132.3.31	tightUnmarshal . . . . .	788
6.132.3.32	tightUnmarshalBrokerError . . . . .	788
6.132.3.33	tightUnmarshalByteArray . . . . .	789
6.132.3.34	tightUnmarshalCachedObject . . . . .	789
6.132.3.35	tightUnmarshalConstByteArray . . . . .	790
6.132.3.36	tightUnmarshalLong . . . . .	790
6.132.3.37	tightUnmarshalNestedObject . . . . .	791
6.132.3.38	tightUnmarshalString . . . . .	791
6.132.3.39	toHexFromBytes . . . . .	791
6.132.3.40	toString . . . . .	792
6.132.3.41	toString . . . . .	792
6.132.3.42	toString . . . . .	792
6.133	activemq::commands::BaseDataStructure Class Reference . . . . .	793
6.133.1	Constructor & Destructor Documentation . . . . .	794
6.133.1.1	~BaseDataStructure . . . . .	794
6.133.2	Member Function Documentation . . . . .	794
6.133.2.1	afterMarshal . . . . .	794
6.133.2.2	afterUnmarshal . . . . .	794
6.133.2.3	beforeMarshal . . . . .	794
6.133.2.4	beforeUnmarshal . . . . .	795
6.133.2.5	copyDataStructure . . . . .	795
6.133.2.6	equals . . . . .	795
6.133.2.7	getMarshaledForm . . . . .	795
6.133.2.8	isMarshalAware . . . . .	796
6.133.2.9	setMarshaledForm . . . . .	796
6.133.2.10	toString . . . . .	796
6.134	binary_function Class Reference . . . . .	797

6.135decaf::net::BindException Class Reference . . . . .	797
6.135.1 Constructor & Destructor Documentation . . . . .	798
6.135.1.1 BindException . . . . .	798
6.135.1.2 BindException . . . . .	798
6.135.1.3 BindException . . . . .	798
6.135.1.4 BindException . . . . .	799
6.135.1.5 BindException . . . . .	799
6.135.1.6 BindException . . . . .	799
6.135.1.7 ~BindException . . . . .	800
6.135.2 Member Function Documentation . . . . .	800
6.135.2.1 clone . . . . .	800
6.136decaf::io::BlockingByteArrayInputStream Class Reference . . . . .	800
6.136.1 Detailed Description . . . . .	801
6.136.2 Constructor & Destructor Documentation . . . . .	801
6.136.2.1 BlockingByteArrayInputStream . . . . .	802
6.136.2.2 BlockingByteArrayInputStream . . . . .	802
6.136.2.3 ~BlockingByteArrayInputStream . . . . .	802
6.136.3 Member Function Documentation . . . . .	802
6.136.3.1 available . . . . .	802
6.136.3.2 close . . . . .	802
6.136.3.3 doReadArrayBounded . . . . .	803
6.136.3.4 doReadByte . . . . .	803
6.136.3.5 setByteArray . . . . .	803
6.136.3.6 skip . . . . .	803
6.137decaf::util::concurrent::BlockingQueue< E > Class Template Reference . . . . .	804
6.137.1 Detailed Description . . . . .	805
6.137.2 Constructor & Destructor Documentation . . . . .	807
6.137.2.1 ~BlockingQueue . . . . .	807
6.137.3 Member Function Documentation . . . . .	807
6.137.3.1 drainTo . . . . .	807
6.137.3.2 drainTo . . . . .	807
6.137.3.3 offer . . . . .	808
6.137.3.4 poll . . . . .	809
6.137.3.5 put . . . . .	809

6.137.3.6 remainingCapacity . . . . .	810
6.137.3.7 take . . . . .	810
6.138decaf::lang::Boolean Class Reference . . . . .	810
6.138.1 Constructor & Destructor Documentation . . . . .	812
6.138.1.1 Boolean . . . . .	812
6.138.1.2 Boolean . . . . .	812
6.138.1.3 ~Boolean . . . . .	812
6.138.2 Member Function Documentation . . . . .	812
6.138.2.1 booleanValue . . . . .	812
6.138.2.2 compareTo . . . . .	812
6.138.2.3 compareTo . . . . .	812
6.138.2.4 equals . . . . .	813
6.138.2.5 equals . . . . .	813
6.138.2.6 operator< . . . . .	813
6.138.2.7 operator< . . . . .	813
6.138.2.8 operator== . . . . .	814
6.138.2.9 operator== . . . . .	814
6.138.2.10parseBoolean . . . . .	814
6.138.2.11toString . . . . .	815
6.138.2.12toString . . . . .	815
6.138.2.13valueOf . . . . .	815
6.138.2.14valueOf . . . . .	815
6.138.3 Field Documentation . . . . .	815
6.138.3.1 _FALSE . . . . .	815
6.138.3.2 _TRUE . . . . .	816
6.139activemq::commands::BooleanExpression Class Reference . . . . .	816
6.139.1 Constructor & Destructor Documentation . . . . .	816
6.139.1.1 BooleanExpression . . . . .	816
6.139.1.2 ~BooleanExpression . . . . .	816
6.139.2 Member Function Documentation . . . . .	816
6.139.2.1 cloneDataStructure . . . . .	817
6.139.2.2 copyDataStructure . . . . .	817
6.139.2.3 equals . . . . .	817
6.139.2.4 toString . . . . .	817

6.140activemq::wireformat::openwire::utils::BooleanStream Class Reference . . . . .	818
6.140.1 Detailed Description . . . . .	818
6.140.2 Constructor & Destructor Documentation . . . . .	819
6.140.2.1 BooleanStream . . . . .	819
6.140.2.2 ~BooleanStream . . . . .	819
6.140.3 Member Function Documentation . . . . .	819
6.140.3.1 clear . . . . .	819
6.140.3.2 marshal . . . . .	819
6.140.3.3 marshal . . . . .	819
6.140.3.4 marshalledSize . . . . .	819
6.140.3.5 readBoolean . . . . .	820
6.140.3.6 unmarshal . . . . .	820
6.140.3.7 writeBoolean . . . . .	820
6.141decaf::util::concurrent::BrokenBarrierException Class Reference . . . . .	820
6.141.1 Constructor & Destructor Documentation . . . . .	821
6.141.1.1 BrokenBarrierException . . . . .	821
6.141.1.2 BrokenBarrierException . . . . .	821
6.141.1.3 BrokenBarrierException . . . . .	821
6.141.1.4 BrokenBarrierException . . . . .	822
6.141.1.5 BrokenBarrierException . . . . .	822
6.141.1.6 BrokenBarrierException . . . . .	822
6.141.1.7 ~BrokenBarrierException . . . . .	822
6.141.2 Member Function Documentation . . . . .	822
6.141.2.1 clone . . . . .	823
6.142activemq::commands::BrokerError Class Reference . . . . .	823
6.142.1 Detailed Description . . . . .	824
6.142.2 Constructor & Destructor Documentation . . . . .	824
6.142.2.1 BrokerError . . . . .	824
6.142.2.2 ~BrokerError . . . . .	824
6.142.3 Member Function Documentation . . . . .	824
6.142.3.1 cloneDataStructure . . . . .	824
6.142.3.2 copyDataStructure . . . . .	825
6.142.3.3 getCause . . . . .	825
6.142.3.4 getDataStructureType . . . . .	825

6.142.3.5	getExceptionClass . . . . .	825
6.142.3.6	getMessage . . . . .	826
6.142.3.7	getStackTraceElements . . . . .	826
6.142.3.8	setCause . . . . .	826
6.142.3.9	setExceptionClass . . . . .	826
6.142.3.10	setMessage . . . . .	826
6.142.3.11	setStackTraceElements . . . . .	827
6.142.3.12	visit . . . . .	827
6.143	activemq::exceptions::BrokerException Class Reference . . . . .	827
6.143.1	Constructor & Destructor Documentation . . . . .	828
6.143.1.1	BrokerException . . . . .	828
6.143.1.2	BrokerException . . . . .	828
6.143.1.3	BrokerException . . . . .	828
6.143.1.4	BrokerException . . . . .	828
6.143.1.5	BrokerException . . . . .	828
6.143.1.6	~BrokerException . . . . .	828
6.143.2	Member Function Documentation . . . . .	828
6.143.2.1	clone . . . . .	828
6.144	activemq::commands::BrokerId Class Reference . . . . .	828
6.144.1	Member Typedef Documentation . . . . .	830
6.144.1.1	COMPARATOR . . . . .	830
6.144.2	Constructor & Destructor Documentation . . . . .	830
6.144.2.1	BrokerId . . . . .	830
6.144.2.2	BrokerId . . . . .	830
6.144.2.3	~BrokerId . . . . .	830
6.144.3	Member Function Documentation . . . . .	830
6.144.3.1	cloneDataStructure . . . . .	830
6.144.3.2	compareTo . . . . .	830
6.144.3.3	copyDataStructure . . . . .	830
6.144.3.4	equals . . . . .	830
6.144.3.5	equals . . . . .	831
6.144.3.6	getDataStructureType . . . . .	831
6.144.3.7	getValue . . . . .	831
6.144.3.8	getValue . . . . .	831

6.144.3.9 operator< . . . . .	831
6.144.3.10 operator= . . . . .	831
6.144.3.11 operator== . . . . .	831
6.144.3.12 setValue . . . . .	831
6.144.3.13 toString . . . . .	831
6.144.4 Field Documentation . . . . .	832
6.144.4.1 ID_BROKERID . . . . .	832
6.144.4.2 value . . . . .	832
6.145 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference . . . . .	832
6.145.1 Detailed Description . . . . .	833
6.145.2 Constructor & Destructor Documentation . . . . .	833
6.145.2.1 BrokerIdMarshaller . . . . .	833
6.145.2.2 ~BrokerIdMarshaller . . . . .	833
6.145.3 Member Function Documentation . . . . .	833
6.145.3.1 createObject . . . . .	833
6.145.3.2 getDataStructureType . . . . .	833
6.145.3.3 looseMarshal . . . . .	834
6.145.3.4 looseUnmarshal . . . . .	834
6.145.3.5 tightMarshal1 . . . . .	834
6.145.3.6 tightMarshal2 . . . . .	835
6.145.3.7 tightUnmarshal . . . . .	835
6.146 activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference . . . . .	836
6.146.1 Detailed Description . . . . .	837
6.146.2 Constructor & Destructor Documentation . . . . .	837
6.146.2.1 BrokerIdMarshaller . . . . .	837
6.146.2.2 ~BrokerIdMarshaller . . . . .	837
6.146.3 Member Function Documentation . . . . .	837
6.146.3.1 createObject . . . . .	837
6.146.3.2 getDataStructureType . . . . .	837
6.146.3.3 looseMarshal . . . . .	838
6.146.3.4 looseUnmarshal . . . . .	838
6.146.3.5 tightMarshal1 . . . . .	838

6.146.3.6	tightMarshal2 . . . . .	839
6.146.3.7	tightUnmarshal . . . . .	839
6.147	activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference . . . . .	840
6.147.1	Detailed Description . . . . .	841
6.147.2	Constructor & Destructor Documentation . . . . .	841
6.147.2.1	BrokerIdMarshaller . . . . .	841
6.147.2.2	~BrokerIdMarshaller . . . . .	841
6.147.3	Member Function Documentation . . . . .	841
6.147.3.1	createObject . . . . .	841
6.147.3.2	getDataStructureType . . . . .	841
6.147.3.3	looseMarshal . . . . .	842
6.147.3.4	looseUnmarshal . . . . .	842
6.147.3.5	tightMarshal1 . . . . .	842
6.147.3.6	tightMarshal2 . . . . .	843
6.147.3.7	tightUnmarshal . . . . .	843
6.148	activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference . . . . .	844
6.148.1	Detailed Description . . . . .	845
6.148.2	Constructor & Destructor Documentation . . . . .	845
6.148.2.1	BrokerIdMarshaller . . . . .	845
6.148.2.2	~BrokerIdMarshaller . . . . .	845
6.148.3	Member Function Documentation . . . . .	845
6.148.3.1	createObject . . . . .	845
6.148.3.2	getDataStructureType . . . . .	845
6.148.3.3	looseMarshal . . . . .	846
6.148.3.4	looseUnmarshal . . . . .	846
6.148.3.5	tightMarshal1 . . . . .	846
6.148.3.6	tightMarshal2 . . . . .	847
6.148.3.7	tightUnmarshal . . . . .	847
6.149	activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller Class Reference . . . . .	848
6.149.1	Detailed Description . . . . .	849
6.149.2	Constructor & Destructor Documentation . . . . .	849
6.149.2.1	BrokerIdMarshaller . . . . .	849

6.149.2.2 ~BrokerIdMarshaller . . . . .	849
6.149.3 Member Function Documentation . . . . .	849
6.149.3.1 createObject . . . . .	849
6.149.3.2 getDataStructureType . . . . .	849
6.149.3.3 looseMarshal . . . . .	850
6.149.3.4 looseUnmarshal . . . . .	850
6.149.3.5 tightMarshal1 . . . . .	850
6.149.3.6 tightMarshal2 . . . . .	851
6.149.3.7 tightUnmarshal . . . . .	851
6.150activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference . . . . .	852
6.150.1 Detailed Description . . . . .	853
6.150.2 Constructor & Destructor Documentation . . . . .	853
6.150.2.1 BrokerIdMarshaller . . . . .	853
6.150.2.2 ~BrokerIdMarshaller . . . . .	853
6.150.3 Member Function Documentation . . . . .	853
6.150.3.1 createObject . . . . .	853
6.150.3.2 getDataStructureType . . . . .	853
6.150.3.3 looseMarshal . . . . .	854
6.150.3.4 looseUnmarshal . . . . .	854
6.150.3.5 tightMarshal1 . . . . .	854
6.150.3.6 tightMarshal2 . . . . .	855
6.150.3.7 tightUnmarshal . . . . .	855
6.151activemq::commands::BrokerInfo Class Reference . . . . .	856
6.151.1 Constructor & Destructor Documentation . . . . .	858
6.151.1.1 BrokerInfo . . . . .	858
6.151.1.2 ~BrokerInfo . . . . .	858
6.151.2 Member Function Documentation . . . . .	858
6.151.2.1 cloneDataStructure . . . . .	858
6.151.2.2 copyDataStructure . . . . .	858
6.151.2.3 equals . . . . .	858
6.151.2.4 getBrokerId . . . . .	859
6.151.2.5 getBrokerId . . . . .	859
6.151.2.6 getBrokerName . . . . .	859



6.151.2.7	getBrokerName	859
6.151.2.8	getBrokerUploadUrl	859
6.151.2.9	getBrokerUploadUrl	859
6.151.2.10	getBrokerURL	859
6.151.2.11	getBrokerURL	859
6.151.2.12	getConnectionId	859
6.151.2.13	getDataStructureType	859
6.151.2.14	getNetworkProperties	859
6.151.2.15	getNetworkProperties	859
6.151.2.16	getPeerBrokerInfos	859
6.151.2.17	getPeerBrokerInfos	860
6.151.2.18	isBrokerInfo	860
6.151.2.19	isDuplexConnection	860
6.151.2.20	isFaultTolerantConfiguration	860
6.151.2.21	isMasterBroker	860
6.151.2.22	isNetworkConnection	860
6.151.2.23	isSlaveBroker	860
6.151.2.24	setBrokerId	860
6.151.2.25	setBrokerName	860
6.151.2.26	setBrokerUploadUrl	860
6.151.2.27	setBrokerURL	860
6.151.2.28	setConnectionId	860
6.151.2.29	setDuplexConnection	860
6.151.2.30	setFaultTolerantConfiguration	860
6.151.2.31	setMasterBroker	861
6.151.2.32	setNetworkConnection	861
6.151.2.33	setNetworkProperties	861
6.151.2.34	setPeerBrokerInfos	861
6.151.2.35	setSlaveBroker	861
6.151.2.36	toString	861
6.151.2.37	visit	861
6.151.3	Field Documentation	861
6.151.3.1	brokerId	861
6.151.3.2	brokerName	862

6.151.3.3 brokerUploadUrl . . . . .	862
6.151.3.4 brokerURL . . . . .	862
6.151.3.5 connectionId . . . . .	862
6.151.3.6 duplexConnection . . . . .	862
6.151.3.7 faultTolerantConfiguration . . . . .	862
6.151.3.8 ID_BROKERINFO . . . . .	862
6.151.3.9 masterBroker . . . . .	862
6.151.3.10networkConnection . . . . .	862
6.151.3.11networkProperties . . . . .	862
6.151.3.12peerBrokerInfos . . . . .	862
6.151.3.13slaveBroker . . . . .	862
6.152activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference . . . . .	862
6.152.1 Detailed Description . . . . .	863
6.152.2 Constructor & Destructor Documentation . . . . .	863
6.152.2.1 BrokerInfoMarshaller . . . . .	863
6.152.2.2 ~BrokerInfoMarshaller . . . . .	864
6.152.3 Member Function Documentation . . . . .	864
6.152.3.1 createObject . . . . .	864
6.152.3.2 getDataStructureType . . . . .	864
6.152.3.3 looseMarshal . . . . .	864
6.152.3.4 looseUnmarshal . . . . .	865
6.152.3.5 tightMarshal1 . . . . .	865
6.152.3.6 tightMarshal2 . . . . .	866
6.152.3.7 tightUnmarshal . . . . .	866
6.153activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference . . . . .	867
6.153.1 Detailed Description . . . . .	867
6.153.2 Constructor & Destructor Documentation . . . . .	868
6.153.2.1 BrokerInfoMarshaller . . . . .	868
6.153.2.2 ~BrokerInfoMarshaller . . . . .	868
6.153.3 Member Function Documentation . . . . .	868
6.153.3.1 createObject . . . . .	868
6.153.3.2 getDataStructureType . . . . .	868

6.153.3.3 looseMarshal . . . . .	868
6.153.3.4 looseUnmarshal . . . . .	869
6.153.3.5 tightMarshal1 . . . . .	869
6.153.3.6 tightMarshal2 . . . . .	870
6.153.3.7 tightUnmarshal . . . . .	870
6.154activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class	
Reference . . . . .	871
6.154.1 Detailed Description . . . . .	871
6.154.2 Constructor & Destructor Documentation . . . . .	872
6.154.2.1 BrokerInfoMarshaller . . . . .	872
6.154.2.2 ~BrokerInfoMarshaller . . . . .	872
6.154.3 Member Function Documentation . . . . .	872
6.154.3.1 createObject . . . . .	872
6.154.3.2 getDataStructureType . . . . .	872
6.154.3.3 looseMarshal . . . . .	872
6.154.3.4 looseUnmarshal . . . . .	873
6.154.3.5 tightMarshal1 . . . . .	873
6.154.3.6 tightMarshal2 . . . . .	874
6.154.3.7 tightUnmarshal . . . . .	874
6.155activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class	
Reference . . . . .	875
6.155.1 Detailed Description . . . . .	875
6.155.2 Constructor & Destructor Documentation . . . . .	876
6.155.2.1 BrokerInfoMarshaller . . . . .	876
6.155.2.2 ~BrokerInfoMarshaller . . . . .	876
6.155.3 Member Function Documentation . . . . .	876
6.155.3.1 createObject . . . . .	876
6.155.3.2 getDataStructureType . . . . .	876
6.155.3.3 looseMarshal . . . . .	876
6.155.3.4 looseUnmarshal . . . . .	877
6.155.3.5 tightMarshal1 . . . . .	877
6.155.3.6 tightMarshal2 . . . . .	878
6.155.3.7 tightUnmarshal . . . . .	878
6.156activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller Class	
Reference . . . . .	879

6.156.1 Detailed Description . . . . .	879
6.156.2 Constructor & Destructor Documentation . . . . .	880
6.156.2.1 BrokerInfoMarshaller . . . . .	880
6.156.2.2 ~BrokerInfoMarshaller . . . . .	880
6.156.3 Member Function Documentation . . . . .	880
6.156.3.1 createObject . . . . .	880
6.156.3.2 getDataStructureType . . . . .	880
6.156.3.3 looseMarshal . . . . .	880
6.156.3.4 looseUnmarshal . . . . .	881
6.156.3.5 tightMarshal1 . . . . .	881
6.156.3.6 tightMarshal2 . . . . .	882
6.156.3.7 tightUnmarshal . . . . .	882
6.157activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference . . . . .	883
6.157.1 Detailed Description . . . . .	883
6.157.2 Constructor & Destructor Documentation . . . . .	884
6.157.2.1 BrokerInfoMarshaller . . . . .	884
6.157.2.2 ~BrokerInfoMarshaller . . . . .	884
6.157.3 Member Function Documentation . . . . .	884
6.157.3.1 createObject . . . . .	884
6.157.3.2 getDataStructureType . . . . .	884
6.157.3.3 looseMarshal . . . . .	884
6.157.3.4 looseUnmarshal . . . . .	885
6.157.3.5 tightMarshal1 . . . . .	885
6.157.3.6 tightMarshal2 . . . . .	886
6.157.3.7 tightUnmarshal . . . . .	886
6.158decaf::nio::Buffer Class Reference . . . . .	887
6.158.1 Detailed Description . . . . .	888
6.158.2 Constructor & Destructor Documentation . . . . .	889
6.158.2.1 Buffer . . . . .	889
6.158.2.2 Buffer . . . . .	889
6.158.2.3 ~Buffer . . . . .	889
6.158.3 Member Function Documentation . . . . .	889
6.158.3.1 capacity . . . . .	889

6.158.3.2 clear . . . . .	890
6.158.3.3 flip . . . . .	890
6.158.3.4 hasRemaining . . . . .	890
6.158.3.5 isReadOnly . . . . .	890
6.158.3.6 limit . . . . .	891
6.158.3.7 limit . . . . .	891
6.158.3.8 mark . . . . .	891
6.158.3.9 position . . . . .	892
6.158.3.10 position . . . . .	892
6.158.3.11 remaining . . . . .	892
6.158.3.12 reset . . . . .	892
6.158.3.13 rewind . . . . .	893
6.158.4 Field Documentation . . . . .	893
6.158.4.1 _capacity . . . . .	893
6.158.4.2 _limit . . . . .	893
6.158.4.3 _mark . . . . .	893
6.158.4.4 _markSet . . . . .	893
6.158.4.5 _position . . . . .	893
6.159decaf::io::BufferedInputStream Class Reference . . . . .	893
6.159.1 Detailed Description . . . . .	895
6.159.2 Constructor & Destructor Documentation . . . . .	895
6.159.2.1 BufferedInputStream . . . . .	896
6.159.2.2 BufferedInputStream . . . . .	896
6.159.2.3 ~BufferedInputStream . . . . .	896
6.159.3 Member Function Documentation . . . . .	896
6.159.3.1 available . . . . .	896
6.159.3.2 close . . . . .	897
6.159.3.3 doReadArrayBounded . . . . .	897
6.159.3.4 doReadByte . . . . .	897
6.159.3.5 mark . . . . .	897
6.159.3.6 markSupported . . . . .	897
6.159.3.7 reset . . . . .	898
6.159.3.8 skip . . . . .	898
6.160decaf::io::BufferedOutputStream Class Reference . . . . .	899

6.160.1 Detailed Description . . . . .	900
6.160.2 Constructor & Destructor Documentation . . . . .	900
6.160.2.1 BufferedOutputStream . . . . .	900
6.160.2.2 BufferedOutputStream . . . . .	900
6.160.2.3 ~BufferedOutputStream . . . . .	900
6.160.3 Member Function Documentation . . . . .	900
6.160.3.1 doWriteArray . . . . .	901
6.160.3.2 doWriteArrayBounded . . . . .	901
6.160.3.3 doWriteByte . . . . .	901
6.160.3.4 flush . . . . .	901
6.161 decaf::internal::nio::BufferFactory Class Reference . . . . .	901
6.161.1 Detailed Description . . . . .	903
6.161.2 Constructor & Destructor Documentation . . . . .	903
6.161.2.1 ~BufferFactory . . . . .	903
6.161.3 Member Function Documentation . . . . .	903
6.161.3.1 createByteBuffer . . . . .	904
6.161.3.2 createByteBuffer . . . . .	904
6.161.3.3 createByteBuffer . . . . .	905
6.161.3.4 createCharBuffer . . . . .	905
6.161.3.5 createCharBuffer . . . . .	905
6.161.3.6 createCharBuffer . . . . .	906
6.161.3.7 createDoubleBuffer . . . . .	906
6.161.3.8 createDoubleBuffer . . . . .	907
6.161.3.9 createDoubleBuffer . . . . .	907
6.161.3.10 createFloatBuffer . . . . .	908
6.161.3.11 createFloatBuffer . . . . .	908
6.161.3.12 createFloatBuffer . . . . .	909
6.161.3.13 createIntBuffer . . . . .	909
6.161.3.14 createIntBuffer . . . . .	910
6.161.3.15 createIntBuffer . . . . .	910
6.161.3.16 createLongBuffer . . . . .	911
6.161.3.17 createLongBuffer . . . . .	911
6.161.3.18 createLongBuffer . . . . .	912
6.161.3.19 createShortBuffer . . . . .	912

6.161.3.20createShortBuffer . . . . .	913
6.161.3.21createShortBuffer . . . . .	913
6.162decaf::nio::BufferOverflowException Class Reference . . . . .	914
6.162.1 Constructor & Destructor Documentation . . . . .	914
6.162.1.1 BufferOverflowException . . . . .	914
6.162.1.2 BufferOverflowException . . . . .	914
6.162.1.3 BufferOverflowException . . . . .	915
6.162.1.4 BufferOverflowException . . . . .	915
6.162.1.5 BufferOverflowException . . . . .	915
6.162.1.6 BufferOverflowException . . . . .	915
6.162.1.7 ~BufferOverflowException . . . . .	916
6.162.2 Member Function Documentation . . . . .	916
6.162.2.1 clone . . . . .	916
6.163decaf::nio::BufferUnderflowException Class Reference . . . . .	916
6.163.1 Constructor & Destructor Documentation . . . . .	917
6.163.1.1 BufferUnderflowException . . . . .	917
6.163.1.2 BufferUnderflowException . . . . .	917
6.163.1.3 BufferUnderflowException . . . . .	917
6.163.1.4 BufferUnderflowException . . . . .	917
6.163.1.5 BufferUnderflowException . . . . .	917
6.163.1.6 BufferUnderflowException . . . . .	918
6.163.1.7 ~BufferUnderflowException . . . . .	918
6.163.2 Member Function Documentation . . . . .	918
6.163.2.1 clone . . . . .	918
6.164decaf::lang::Byte Class Reference . . . . .	918
6.164.1 Constructor & Destructor Documentation . . . . .	920
6.164.1.1 Byte . . . . .	920
6.164.1.2 Byte . . . . .	920
6.164.1.3 ~Byte . . . . .	920
6.164.2 Member Function Documentation . . . . .	920
6.164.2.1 byteValue . . . . .	921
6.164.2.2 compareTo . . . . .	921
6.164.2.3 compareTo . . . . .	921
6.164.2.4 decode . . . . .	921

6.164.2.5 doubleValue . . . . .	922
6.164.2.6 equals . . . . .	922
6.164.2.7 equals . . . . .	922
6.164.2.8 floatValue . . . . .	923
6.164.2.9 intValue . . . . .	923
6.164.2.10 longValue . . . . .	923
6.164.2.11 operator< . . . . .	923
6.164.2.12 operator< . . . . .	924
6.164.2.13 operator== . . . . .	924
6.164.2.14 operator== . . . . .	924
6.164.2.15 parseByte . . . . .	925
6.164.2.16 parseByte . . . . .	925
6.164.2.17 shortValue . . . . .	926
6.164.2.18 toString . . . . .	926
6.164.2.19 toString . . . . .	926
6.164.2.20 valueOf . . . . .	926
6.164.2.21 valueOf . . . . .	926
6.164.2.22 valueOf . . . . .	927
6.164.3 Field Documentation . . . . .	927
6.164.3.1 MAX_VALUE . . . . .	927
6.164.3.2 MIN_VALUE . . . . .	928
6.164.3.3 SIZE . . . . .	928
6.165 decaf::internal::util::ByteArrayAdapter Class Reference . . . . .	928
6.165.1 Detailed Description . . . . .	932
6.165.2 Constructor & Destructor Documentation . . . . .	932
6.165.2.1 ByteArrayAdapter . . . . .	932
6.165.2.2 ByteArrayAdapter . . . . .	932
6.165.2.3 ByteArrayAdapter . . . . .	933
6.165.2.4 ByteArrayAdapter . . . . .	933
6.165.2.5 ByteArrayAdapter . . . . .	934
6.165.2.6 ByteArrayAdapter . . . . .	934
6.165.2.7 ByteArrayAdapter . . . . .	934
6.165.2.8 ByteArrayAdapter . . . . .	935
6.165.2.9 ~ByteArrayAdapter . . . . .	935



6.165.3 Member Function Documentation . . . . .	935
6.165.3.1 clear . . . . .	935
6.165.3.2 get . . . . .	935
6.165.3.3 getByteArray . . . . .	936
6.165.3.4 getCapacity . . . . .	936
6.165.3.5 getChar . . . . .	936
6.165.3.6 getCharArray . . . . .	937
6.165.3.7 getCharCapacity . . . . .	937
6.165.3.8 getDouble . . . . .	937
6.165.3.9 getDoubleArray . . . . .	938
6.165.3.10 getDoubleAt . . . . .	938
6.165.3.11 getDoubleCapacity . . . . .	938
6.165.3.12 getFloat . . . . .	938
6.165.3.13 getFloatArray . . . . .	939
6.165.3.14 getFloatAt . . . . .	939
6.165.3.15 getFloatCapacity . . . . .	939
6.165.3.16 getInt . . . . .	940
6.165.3.17 getIntArray . . . . .	940
6.165.3.18 getIntAt . . . . .	940
6.165.3.19 getIntCapacity . . . . .	941
6.165.3.20 getLong . . . . .	941
6.165.3.21 getLongArray . . . . .	941
6.165.3.22 getLongAt . . . . .	942
6.165.3.23 getLongCapacity . . . . .	942
6.165.3.24 getShort . . . . .	942
6.165.3.25 getShortArray . . . . .	943
6.165.3.26 getShortAt . . . . .	943
6.165.3.27 getShortCapacity . . . . .	943
6.165.3.28 operator[] . . . . .	943
6.165.3.29 operator[] . . . . .	944
6.165.3.30 put . . . . .	944
6.165.3.31 putChar . . . . .	944
6.165.3.32 putDouble . . . . .	945
6.165.3.33 putDoubleAt . . . . .	945

6.165.3.34	putFloat . . . . .	946
6.165.3.35	putFloatAt . . . . .	946
6.165.3.36	putInt . . . . .	947
6.165.3.37	putIntAt . . . . .	947
6.165.3.38	putLong . . . . .	948
6.165.3.39	putLongAt . . . . .	948
6.165.3.40	putShort . . . . .	949
6.165.3.41	putShortAt . . . . .	949
6.165.3.42	read . . . . .	950
6.165.3.43	resize . . . . .	950
6.165.3.44	write . . . . .	951
6.166	decaf::internal::nio::ByteBuffer Class Reference . . . . .	951
6.166.1	Detailed Description . . . . .	963
6.166.2	Constructor & Destructor Documentation . . . . .	964
6.166.2.1	ByteBuffer . . . . .	964
6.166.2.2	ByteBuffer . . . . .	965
6.166.2.3	ByteBuffer . . . . .	965
6.166.2.4	ByteBuffer . . . . .	966
6.166.2.5	~ByteBuffer . . . . .	966
6.166.3	Member Function Documentation . . . . .	966
6.166.3.1	array . . . . .	966
6.166.3.2	arrayOffset . . . . .	966
6.166.3.3	asCharBuffer . . . . .	967
6.166.3.4	asDoubleBuffer . . . . .	967
6.166.3.5	asFloatBuffer . . . . .	968
6.166.3.6	asIntBuffer . . . . .	968
6.166.3.7	asLongBuffer . . . . .	968
6.166.3.8	asReadOnlyBuffer . . . . .	969
6.166.3.9	asShortBuffer . . . . .	969
6.166.3.10	compact . . . . .	970
6.166.3.11	duplicate . . . . .	970
6.166.3.12	get . . . . .	970
6.166.3.13	get . . . . .	971
6.166.3.14	getChar . . . . .	971

6.166.3.15	getChar . . . . .	972
6.166.3.16	getDouble . . . . .	972
6.166.3.17	getDouble . . . . .	972
6.166.3.18	getFloat . . . . .	973
6.166.3.19	getFloat . . . . .	973
6.166.3.20	getInt . . . . .	974
6.166.3.21	getInt . . . . .	974
6.166.3.22	getLong . . . . .	974
6.166.3.23	getLong . . . . .	975
6.166.3.24	getShort . . . . .	975
6.166.3.25	getShort . . . . .	976
6.166.3.26	hasArray . . . . .	976
6.166.3.27	isReadOnly . . . . .	976
6.166.3.28	put . . . . .	976
6.166.3.29	put . . . . .	977
6.166.3.30	putChar . . . . .	977
6.166.3.31	putChar . . . . .	978
6.166.3.32	putDouble . . . . .	978
6.166.3.33	putDouble . . . . .	979
6.166.3.34	putFloat . . . . .	979
6.166.3.35	putFloat . . . . .	980
6.166.3.36	putInt . . . . .	980
6.166.3.37	putInt . . . . .	981
6.166.3.38	putLong . . . . .	981
6.166.3.39	putLong . . . . .	982
6.166.3.40	putShort . . . . .	982
6.166.3.41	putShort . . . . .	983
6.166.3.42	setReadOnly . . . . .	983
6.166.3.43	slice . . . . .	984
6.167	decaf::io::ByteArrayInputStream Class Reference . . . . .	984
6.167.1	Detailed Description . . . . .	987
6.167.2	Constructor & Destructor Documentation . . . . .	987
6.167.2.1	ByteArrayInputStream . . . . .	987
6.167.2.2	ByteArrayInputStream . . . . .	987

6.167.2.3	ByteArrayInputStream	987
6.167.2.4	ByteArrayInputStream	988
6.167.2.5	~ByteArrayInputStream	988
6.167.3	Member Function Documentation	988
6.167.3.1	available	988
6.167.3.2	doReadArrayBounded	989
6.167.3.3	doReadByte	989
6.167.3.4	mark	989
6.167.3.5	markSupported	989
6.167.3.6	reset	990
6.167.3.7	setByteArray	990
6.167.3.8	setByteArray	991
6.167.3.9	setByteArray	991
6.167.3.10	skip	991
6.168	decaf::nio::ByteArrayOutputStream Class Reference	992
6.168.1	Constructor & Destructor Documentation	993
6.168.1.1	ByteArrayOutputStream	993
6.168.1.2	ByteArrayOutputStream	993
6.168.1.3	~ByteArrayOutputStream	993
6.168.2	Member Function Documentation	993
6.168.2.1	doWriteArrayBounded	993
6.168.2.2	doWriteByte	994
6.168.2.3	reset	994
6.168.2.4	size	994
6.168.2.5	toByteArray	994
6.168.2.6	toString	994
6.168.2.7	writeTo	995
6.169	decaf::nio::ByteBuffer Class Reference	995
6.169.1	Detailed Description	999
6.169.2	Constructor & Destructor Documentation	1000
6.169.2.1	ByteBuffer	1000
6.169.2.2	~ByteBuffer	1000
6.169.3	Member Function Documentation	1000
6.169.3.1	allocate	1000

6.169.3.2 array . . . . .	1001
6.169.3.3 arrayOffset . . . . .	1001
6.169.3.4 asCharBuffer . . . . .	1002
6.169.3.5 asDoubleBuffer . . . . .	1002
6.169.3.6 asFloatBuffer . . . . .	1002
6.169.3.7 asIntBuffer . . . . .	1003
6.169.3.8 asLongBuffer . . . . .	1003
6.169.3.9 asReadOnlyBuffer . . . . .	1003
6.169.3.10asShortBuffer . . . . .	1004
6.169.3.11compact . . . . .	1004
6.169.3.12compareTo . . . . .	1005
6.169.3.13duplicate . . . . .	1005
6.169.3.14equals . . . . .	1005
6.169.3.15get . . . . .	1005
6.169.3.16get . . . . .	1006
6.169.3.17get . . . . .	1006
6.169.3.18get . . . . .	1006
6.169.3.19getChar . . . . .	1007
6.169.3.20getChar . . . . .	1007
6.169.3.21getDouble . . . . .	1008
6.169.3.22getDouble . . . . .	1008
6.169.3.23getFloat . . . . .	1009
6.169.3.24getFloat . . . . .	1009
6.169.3.25getInt . . . . .	1009
6.169.3.26getInt . . . . .	1010
6.169.3.27getLong . . . . .	1010
6.169.3.28getLong . . . . .	1011
6.169.3.29getShort . . . . .	1011
6.169.3.30getShort . . . . .	1011
6.169.3.31hasArray . . . . .	1012
6.169.3.32isReadOnly . . . . .	1012
6.169.3.33operator< . . . . .	1012
6.169.3.34operator== . . . . .	1012
6.169.3.35put . . . . .	1012

6.169.3.36put . . . . .	1013
6.169.3.37put . . . . .	1013
6.169.3.38put . . . . .	1014
6.169.3.39put . . . . .	1015
6.169.3.40putChar . . . . .	1015
6.169.3.41putChar . . . . .	1016
6.169.3.42putDouble . . . . .	1016
6.169.3.43putDouble . . . . .	1017
6.169.3.44putFloat . . . . .	1017
6.169.3.45putFloat . . . . .	1018
6.169.3.46putInt . . . . .	1018
6.169.3.47putInt . . . . .	1019
6.169.3.48putLong . . . . .	1019
6.169.3.49putLong . . . . .	1020
6.169.3.50putShort . . . . .	1020
6.169.3.51putShort . . . . .	1021
6.169.3.52slice . . . . .	1021
6.169.3.53toString . . . . .	1022
6.169.3.54wrap . . . . .	1022
6.169.3.55wrap . . . . .	1022
6.170cms::BytesMessage Class Reference . . . . .	1023
6.170.1 Detailed Description . . . . .	1025
6.170.2 Constructor & Destructor Documentation . . . . .	1026
6.170.2.1 ~BytesMessage . . . . .	1026
6.170.3 Member Function Documentation . . . . .	1026
6.170.3.1 clone . . . . .	1026
6.170.3.2 getBodyBytes . . . . .	1027
6.170.3.3 getBodyLength . . . . .	1027
6.170.3.4 readBoolean . . . . .	1027
6.170.3.5 readByte . . . . .	1028
6.170.3.6 readBytes . . . . .	1028
6.170.3.7 readBytes . . . . .	1029
6.170.3.8 readChar . . . . .	1030
6.170.3.9 readDouble . . . . .	1030

6.170.3.10	readFloat . . . . .	1031
6.170.3.11	readInt . . . . .	1031
6.170.3.12	readLong . . . . .	1032
6.170.3.13	readShort . . . . .	1032
6.170.3.14	readString . . . . .	1033
6.170.3.15	readUnsignedShort . . . . .	1033
6.170.3.16	readUTF . . . . .	1034
6.170.3.17	reset . . . . .	1034
6.170.3.18	setBodyBytes . . . . .	1034
6.170.3.19	writeBoolean . . . . .	1035
6.170.3.20	writeByte . . . . .	1035
6.170.3.21	writeBytes . . . . .	1036
6.170.3.22	writeBytes . . . . .	1036
6.170.3.23	writeChar . . . . .	1037
6.170.3.24	writeDouble . . . . .	1037
6.170.3.25	writeFloat . . . . .	1038
6.170.3.26	writeInt . . . . .	1038
6.170.3.27	writeLong . . . . .	1038
6.170.3.28	writeShort . . . . .	1039
6.170.3.29	writeString . . . . .	1039
6.170.3.30	writeUnsignedShort . . . . .	1040
6.170.3.31	writeUTF . . . . .	1040
6.171	activemq::cmsutil::CachedConsumer Class Reference . . . . .	1041
6.171.1	Detailed Description . . . . .	1041
6.171.2	Constructor & Destructor Documentation . . . . .	1042
6.171.2.1	CachedConsumer . . . . .	1042
6.171.2.2	CachedConsumer . . . . .	1042
6.171.2.3	~CachedConsumer . . . . .	1042
6.171.3	Member Function Documentation . . . . .	1042
6.171.3.1	close . . . . .	1042
6.171.3.2	getMessageListener . . . . .	1042
6.171.3.3	getMessageSelector . . . . .	1042
6.171.3.4	operator= . . . . .	1043
6.171.3.5	receive . . . . .	1043

6.171.3.6 receive . . . . .	1043
6.171.3.7 receiveNoWait . . . . .	1043
6.171.3.8 setMessageListener . . . . .	1044
6.172activemq::cmsutil::CachedProducer Class Reference . . . . .	1044
6.172.1 Detailed Description . . . . .	1045
6.172.2 Constructor & Destructor Documentation . . . . .	1045
6.172.2.1 CachedProducer . . . . .	1045
6.172.2.2 CachedProducer . . . . .	1046
6.172.2.3 ~CachedProducer . . . . .	1046
6.172.3 Member Function Documentation . . . . .	1046
6.172.3.1 close . . . . .	1046
6.172.3.2 getDeliveryMode . . . . .	1046
6.172.3.3 getDisableMessageID . . . . .	1046
6.172.3.4 getDisableMessageTimeStamp . . . . .	1047
6.172.3.5 getPriority . . . . .	1047
6.172.3.6 getTimeToLive . . . . .	1047
6.172.3.7 operator= . . . . .	1047
6.172.3.8 send . . . . .	1048
6.172.3.9 send . . . . .	1048
6.172.3.10send . . . . .	1049
6.172.3.11send . . . . .	1049
6.172.3.12setDeliveryMode . . . . .	1050
6.172.3.13setDisableMessageID . . . . .	1050
6.172.3.14setDisableMessageTimeStamp . . . . .	1050
6.172.3.15setPriority . . . . .	1051
6.172.3.16setTimeToLive . . . . .	1051
6.173decaf::util::concurrent::Callable< V > Class Template Reference . . . . .	1051
6.173.1 Detailed Description . . . . .	1052
6.173.2 Constructor & Destructor Documentation . . . . .	1052
6.173.2.1 ~Callable . . . . .	1052
6.173.3 Member Function Documentation . . . . .	1052
6.173.3.1 call . . . . .	1052
6.174decaf::util::concurrent::CancellationException Class Reference . . . . .	1052
6.174.1 Constructor & Destructor Documentation . . . . .	1053



6.174.1.1 CancellationException . . . . .	1053
6.174.1.2 CancellationException . . . . .	1053
6.174.1.3 CancellationException . . . . .	1053
6.174.1.4 CancellationException . . . . .	1054
6.174.1.5 CancellationException . . . . .	1054
6.174.1.6 CancellationException . . . . .	1054
6.174.1.7 ~CancellationException . . . . .	1055
6.174.2 Member Function Documentation . . . . .	1055
6.174.2.1 clone . . . . .	1055
6.175decaf::security::cert::Certificate Class Reference . . . . .	1055
6.175.1 Detailed Description . . . . .	1056
6.175.2 Constructor & Destructor Documentation . . . . .	1056
6.175.2.1 ~Certificate . . . . .	1056
6.175.3 Member Function Documentation . . . . .	1056
6.175.3.1 equals . . . . .	1056
6.175.3.2 getEncoded . . . . .	1056
6.175.3.3 getPublicKey . . . . .	1057
6.175.3.4 getPublicKey . . . . .	1057
6.175.3.5 getType . . . . .	1057
6.175.3.6 toString . . . . .	1057
6.175.3.7 verify . . . . .	1058
6.175.3.8 verify . . . . .	1058
6.176decaf::security::cert::CertificateEncodingException Class Reference . . . . .	1059
6.176.1 Constructor & Destructor Documentation . . . . .	1060
6.176.1.1 CertificateEncodingException . . . . .	1060
6.176.1.2 CertificateEncodingException . . . . .	1060
6.176.1.3 CertificateEncodingException . . . . .	1060
6.176.1.4 CertificateEncodingException . . . . .	1060
6.176.1.5 ~CertificateEncodingException . . . . .	1060
6.176.2 Member Function Documentation . . . . .	1060
6.176.2.1 clone . . . . .	1061
6.177decaf::security::cert::CertificateException Class Reference . . . . .	1061
6.177.1 Constructor & Destructor Documentation . . . . .	1061
6.177.1.1 CertificateException . . . . .	1061

6.177.1.2 CertificateException . . . . .	1062
6.177.1.3 CertificateException . . . . .	1062
6.177.1.4 CertificateException . . . . .	1062
6.177.1.5 ~CertificateException . . . . .	1062
6.177.2 Member Function Documentation . . . . .	1062
6.177.2.1 clone . . . . .	1062
6.178decaf::security::cert::CertificateExpiredException Class Reference . . .	1063
6.178.1 Constructor & Destructor Documentation . . . . .	1063
6.178.1.1 CertificateExpiredException . . . . .	1063
6.178.1.2 CertificateExpiredException . . . . .	1064
6.178.1.3 CertificateExpiredException . . . . .	1064
6.178.1.4 CertificateExpiredException . . . . .	1064
6.178.1.5 ~CertificateExpiredException . . . . .	1064
6.178.2 Member Function Documentation . . . . .	1064
6.178.2.1 clone . . . . .	1064
6.179decaf::security::cert::CertificateNotYetValidException Class Reference .	1065
6.179.1 Constructor & Destructor Documentation . . . . .	1065
6.179.1.1 CertificateNotYetValidException . . . . .	1065
6.179.1.2 CertificateNotYetValidException . . . . .	1066
6.179.1.3 CertificateNotYetValidException . . . . .	1066
6.179.1.4 CertificateNotYetValidException . . . . .	1066
6.179.1.5 ~CertificateNotYetValidException . . . . .	1066
6.179.2 Member Function Documentation . . . . .	1066
6.179.2.1 clone . . . . .	1066
6.180decaf::security::cert::CertificateParsingException Class Reference . . .	1067
6.180.1 Constructor & Destructor Documentation . . . . .	1067
6.180.1.1 CertificateParsingException . . . . .	1067
6.180.1.2 CertificateParsingException . . . . .	1068
6.180.1.3 CertificateParsingException . . . . .	1068
6.180.1.4 CertificateParsingException . . . . .	1068
6.180.1.5 ~CertificateParsingException . . . . .	1068
6.180.2 Member Function Documentation . . . . .	1068
6.180.2.1 clone . . . . .	1068
6.181decaf::lang::Character Class Reference . . . . .	1069

6.181.1 Constructor & Destructor Documentation . . . . .	1071
6.181.1.1 Character . . . . .	1071
6.181.2 Member Function Documentation . . . . .	1071
6.181.2.1 byteValue . . . . .	1071
6.181.2.2 compareTo . . . . .	1071
6.181.2.3 compareTo . . . . .	1071
6.181.2.4 digit . . . . .	1072
6.181.2.5 doubleValue . . . . .	1072
6.181.2.6 equals . . . . .	1072
6.181.2.7 equals . . . . .	1073
6.181.2.8 floatValue . . . . .	1073
6.181.2.9 intValue . . . . .	1073
6.181.2.10 isDigit . . . . .	1073
6.181.2.11 isISOControl . . . . .	1073
6.181.2.12 isLetter . . . . .	1074
6.181.2.13 isLetterOrDigit . . . . .	1074
6.181.2.14 isLowerCase . . . . .	1074
6.181.2.15 isUpperCase . . . . .	1074
6.181.2.16 isWhitespace . . . . .	1074
6.181.2.17 longValue . . . . .	1074
6.181.2.18 operator< . . . . .	1074
6.181.2.19 operator< . . . . .	1075
6.181.2.20 operator== . . . . .	1075
6.181.2.21 operator== . . . . .	1075
6.181.2.22 shortValue . . . . .	1076
6.181.2.23 toString . . . . .	1076
6.181.2.24 valueOf . . . . .	1076
6.181.3 Field Documentation . . . . .	1076
6.181.3.1 MAX_RADIX . . . . .	1076
6.181.3.2 MAX_VALUE . . . . .	1076
6.181.3.3 MIN_RADIX . . . . .	1076
6.181.3.4 MIN_VALUE . . . . .	1077
6.181.3.5 SIZE . . . . .	1077
6.182 decaf::internal::nio::CharArrayBuffer Class Reference . . . . .	1077

6.182.1 Constructor & Destructor Documentation . . . . .	1081
6.182.1.1 CharArrayBuffer . . . . .	1081
6.182.1.2 CharArrayBuffer . . . . .	1082
6.182.1.3 CharArrayBuffer . . . . .	1082
6.182.1.4 CharArrayBuffer . . . . .	1083
6.182.1.5 ~CharArrayBuffer . . . . .	1083
6.182.2 Member Function Documentation . . . . .	1083
6.182.2.1 array . . . . .	1083
6.182.2.2 arrayOffset . . . . .	1083
6.182.2.3 asReadOnlyBuffer . . . . .	1084
6.182.2.4 compact . . . . .	1084
6.182.2.5 duplicate . . . . .	1085
6.182.2.6 get . . . . .	1085
6.182.2.7 get . . . . .	1085
6.182.2.8 hasArray . . . . .	1086
6.182.2.9 isReadOnly . . . . .	1086
6.182.2.10put . . . . .	1086
6.182.2.11put . . . . .	1087
6.182.2.12setReadOnly . . . . .	1087
6.182.2.13slice . . . . .	1088
6.182.2.14subSequence . . . . .	1088
6.182.3 Field Documentation . . . . .	1089
6.182.3.1 _array . . . . .	1089
6.182.3.2 length . . . . .	1089
6.182.3.3 offset . . . . .	1089
6.182.3.4 readOnly . . . . .	1089
6.183decaf::nio::CharBuffer Class Reference . . . . .	1089
6.183.1 Detailed Description . . . . .	1092
6.183.2 Constructor & Destructor Documentation . . . . .	1092
6.183.2.1 CharBuffer . . . . .	1092
6.183.2.2 ~CharBuffer . . . . .	1092
6.183.3 Member Function Documentation . . . . .	1092
6.183.3.1 allocate . . . . .	1093
6.183.3.2 append . . . . .	1093

6.183.3.3 append . . . . .	1094
6.183.3.4 append . . . . .	1094
6.183.3.5 array . . . . .	1095
6.183.3.6 arrayOffset . . . . .	1095
6.183.3.7 asReadOnlyBuffer . . . . .	1096
6.183.3.8 charAt . . . . .	1096
6.183.3.9 compact . . . . .	1096
6.183.3.10compareTo . . . . .	1097
6.183.3.11duplicate . . . . .	1097
6.183.3.12equals . . . . .	1097
6.183.3.13get . . . . .	1097
6.183.3.14get . . . . .	1098
6.183.3.15get . . . . .	1098
6.183.3.16get . . . . .	1099
6.183.3.17hasArray . . . . .	1099
6.183.3.18length . . . . .	1100
6.183.3.19operator< . . . . .	1100
6.183.3.20operator== . . . . .	1100
6.183.3.21put . . . . .	1100
6.183.3.22put . . . . .	1101
6.183.3.23put . . . . .	1101
6.183.3.24put . . . . .	1102
6.183.3.25put . . . . .	1102
6.183.3.26put . . . . .	1103
6.183.3.27put . . . . .	1104
6.183.3.28read . . . . .	1104
6.183.3.29slice . . . . .	1105
6.183.3.30subSequence . . . . .	1105
6.183.3.31toString . . . . .	1106
6.183.3.32wrap . . . . .	1106
6.183.3.33wrap . . . . .	1107
6.184decaf::lang::CharSequence Class Reference . . . . .	1107
6.184.1 Detailed Description . . . . .	1108
6.184.2 Constructor & Destructor Documentation . . . . .	1108

6.184.2.1 ~CharSequence . . . . .	1108
6.184.3 Member Function Documentation . . . . .	1108
6.184.3.1 charAt . . . . .	1108
6.184.3.2 length . . . . .	1108
6.184.3.3 subSequence . . . . .	1109
6.184.3.4 toString . . . . .	1109
6.185decaf::util::zip::CheckedInputStream Class Reference . . . . .	1109
6.185.1 Detailed Description . . . . .	1110
6.185.2 Constructor & Destructor Documentation . . . . .	1110
6.185.2.1 CheckedInputStream . . . . .	1111
6.185.2.2 ~CheckedInputStream . . . . .	1111
6.185.3 Member Function Documentation . . . . .	1111
6.185.3.1 doReadArrayBounded . . . . .	1111
6.185.3.2 doReadByte . . . . .	1111
6.185.3.3 getChecksum . . . . .	1111
6.185.3.4 skip . . . . .	1112
6.186decaf::util::zip::CheckedOutputStream Class Reference . . . . .	1112
6.186.1 Detailed Description . . . . .	1113
6.186.2 Constructor & Destructor Documentation . . . . .	1113
6.186.2.1 CheckedOutputStream . . . . .	1113
6.186.2.2 ~CheckedOutputStream . . . . .	1113
6.186.3 Member Function Documentation . . . . .	1113
6.186.3.1 doWriteArrayBounded . . . . .	1114
6.186.3.2 doWriteByte . . . . .	1114
6.186.3.3 getChecksum . . . . .	1114
6.187decaf::util::zip::Checksum Class Reference . . . . .	1114
6.187.1 Detailed Description . . . . .	1115
6.187.2 Constructor & Destructor Documentation . . . . .	1115
6.187.2.1 ~Checksum . . . . .	1115
6.187.3 Member Function Documentation . . . . .	1115
6.187.3.1 getValue . . . . .	1115
6.187.3.2 reset . . . . .	1115
6.187.3.3 update . . . . .	1115
6.187.3.4 update . . . . .	1116

6.187.3.5 update . . . . .	1116
6.187.3.6 update . . . . .	1116
6.188decaf::lang::exceptions::ClassCastException Class Reference . . . . .	1117
6.188.1 Constructor & Destructor Documentation . . . . .	1117
6.188.1.1 ClassCastException . . . . .	1117
6.188.1.2 ClassCastException . . . . .	1117
6.188.1.3 ClassCastException . . . . .	1118
6.188.1.4 ClassCastException . . . . .	1118
6.188.1.5 ClassCastException . . . . .	1118
6.188.1.6 ClassCastException . . . . .	1118
6.188.1.7 ~ClassCastException . . . . .	1119
6.188.2 Member Function Documentation . . . . .	1119
6.188.2.1 clone . . . . .	1119
6.189cms::Closeable Class Reference . . . . .	1119
6.189.1 Detailed Description . . . . .	1120
6.189.2 Constructor & Destructor Documentation . . . . .	1120
6.189.2.1 ~Closeable . . . . .	1120
6.189.3 Member Function Documentation . . . . .	1120
6.189.3.1 close . . . . .	1120
6.190decaf::io::Closeable Class Reference . . . . .	1120
6.190.1 Detailed Description . . . . .	1121
6.190.2 Constructor & Destructor Documentation . . . . .	1121
6.190.2.1 ~Closeable . . . . .	1121
6.190.3 Member Function Documentation . . . . .	1121
6.190.3.1 close . . . . .	1121
6.191activemq::transport::failover::CloseTransportsTask Class Reference . . . . .	1122
6.191.1 Constructor & Destructor Documentation . . . . .	1122
6.191.1.1 CloseTransportsTask . . . . .	1122
6.191.1.2 ~CloseTransportsTask . . . . .	1122
6.191.2 Member Function Documentation . . . . .	1122
6.191.2.1 add . . . . .	1122
6.191.2.2 isPending . . . . .	1122
6.191.2.3 iterate . . . . .	1123
6.192activemq::cmsutil::CmsAccessor Class Reference . . . . .	1123

6.192.1 Detailed Description . . . . .	1124
6.192.2 Constructor & Destructor Documentation . . . . .	1124
6.192.2.1 CmsAccessor . . . . .	1124
6.192.2.2 CmsAccessor . . . . .	1124
6.192.2.3 ~CmsAccessor . . . . .	1124
6.192.3 Member Function Documentation . . . . .	1124
6.192.3.1 checkConnectionFactory . . . . .	1125
6.192.3.2 createConnection . . . . .	1125
6.192.3.3 createSession . . . . .	1125
6.192.3.4 destroy . . . . .	1126
6.192.3.5 getConnectionFactory . . . . .	1126
6.192.3.6 getConnectionFactory . . . . .	1126
6.192.3.7 getResourceLifecycleManager . . . . .	1126
6.192.3.8 getResourceLifecycleManager . . . . .	1126
6.192.3.9 getSessionAcknowledgeMode . . . . .	1126
6.192.3.10 init . . . . .	1126
6.192.3.11 operator= . . . . .	1127
6.192.3.12 setConnectionFactory . . . . .	1127
6.192.3.13 setSessionAcknowledgeMode . . . . .	1127
6.193activemq::cmsutil::CmsDestinationAccessor Class Reference . . . . .	1127
6.193.1 Detailed Description . . . . .	1128
6.193.2 Constructor & Destructor Documentation . . . . .	1128
6.193.2.1 CmsDestinationAccessor . . . . .	1128
6.193.2.2 CmsDestinationAccessor . . . . .	1128
6.193.2.3 ~CmsDestinationAccessor . . . . .	1128
6.193.3 Member Function Documentation . . . . .	1128
6.193.3.1 checkDestinationResolver . . . . .	1129
6.193.3.2 destroy . . . . .	1129
6.193.3.3 getDestinationResolver . . . . .	1129
6.193.3.4 getDestinationResolver . . . . .	1129
6.193.3.5 init . . . . .	1129
6.193.3.6 isPubSubDomain . . . . .	1129
6.193.3.7 operator= . . . . .	1129
6.193.3.8 resolveDestinationName . . . . .	1129



6.193.3.9 setDestinationResolver . . . . .	1130
6.193.3.10setPubSubDomain . . . . .	1130
6.194cms::CMSEException Class Reference . . . . .	1130
6.194.1 Detailed Description . . . . .	1131
6.194.2 Constructor & Destructor Documentation . . . . .	1131
6.194.2.1 CMSEException . . . . .	1131
6.194.2.2 CMSEException . . . . .	1132
6.194.2.3 CMSEException . . . . .	1132
6.194.2.4 CMSEException . . . . .	1132
6.194.2.5 ~CMSEException . . . . .	1132
6.194.3 Member Function Documentation . . . . .	1132
6.194.3.1 getCause . . . . .	1132
6.194.3.2 getMessage . . . . .	1132
6.194.3.3 getStackTrace . . . . .	1132
6.194.3.4 getStackTraceString . . . . .	1133
6.194.3.5 printStackTrace . . . . .	1133
6.194.3.6 printStackTrace . . . . .	1133
6.194.3.7 setMark . . . . .	1133
6.194.3.8 what . . . . .	1133
6.195activemq::util::CMSEExceptionSupport Class Reference . . . . .	1134
6.195.1 Constructor & Destructor Documentation . . . . .	1134
6.195.1.1 ~CMSEExceptionSupport . . . . .	1134
6.195.2 Member Function Documentation . . . . .	1134
6.195.2.1 create . . . . .	1134
6.195.2.2 create . . . . .	1134
6.195.2.3 createMessageEOFException . . . . .	1134
6.195.2.4 createMessageFormatException . . . . .	1134
6.196cms::CMSProperties Class Reference . . . . .	1135
6.196.1 Detailed Description . . . . .	1136
6.196.2 Constructor & Destructor Documentation . . . . .	1136
6.196.2.1 ~CMSProperties . . . . .	1136
6.196.3 Member Function Documentation . . . . .	1136
6.196.3.1 clear . . . . .	1136
6.196.3.2 clone . . . . .	1136

6.196.3.3 copy . . . . .	1136
6.196.3.4 getProperty . . . . .	1136
6.196.3.5 getProperty . . . . .	1137
6.196.3.6 hasProperty . . . . .	1137
6.196.3.7 isEmpty . . . . .	1137
6.196.3.8 remove . . . . .	1138
6.196.3.9 setProperty . . . . .	1138
6.196.3.10toArray . . . . .	1138
6.196.3.11toString . . . . .	1138
6.197cms::CMSSecurityException Class Reference . . . . .	1139
6.197.1 Detailed Description . . . . .	1139
6.197.2 Constructor & Destructor Documentation . . . . .	1139
6.197.2.1 CMSSecurityException . . . . .	1139
6.197.2.2 CMSSecurityException . . . . .	1139
6.197.2.3 CMSSecurityException . . . . .	1139
6.197.2.4 CMSSecurityException . . . . .	1139
6.197.2.5 ~CMSSecurityException . . . . .	1140
6.198activemq::cmsutil::CmsTemplate Class Reference . . . . .	1140
6.198.1 Detailed Description . . . . .	1143
6.198.2 Constructor & Destructor Documentation . . . . .	1143
6.198.2.1 CmsTemplate . . . . .	1143
6.198.2.2 CmsTemplate . . . . .	1143
6.198.2.3 CmsTemplate . . . . .	1143
6.198.2.4 ~CmsTemplate . . . . .	1143
6.198.3 Member Function Documentation . . . . .	1143
6.198.3.1 destroy . . . . .	1144
6.198.3.2 execute . . . . .	1144
6.198.3.3 execute . . . . .	1144
6.198.3.4 execute . . . . .	1144
6.198.3.5 execute . . . . .	1145
6.198.3.6 getDefaultDestination . . . . .	1145
6.198.3.7 getDefaultDestination . . . . .	1145
6.198.3.8 getDefaultDestinationName . . . . .	1145
6.198.3.9 getDeliveryMode . . . . .	1146

6.198.3.10getPriority . . . . .	1146
6.198.3.11getReceiveTimeout . . . . .	1146
6.198.3.12getTimeToLive . . . . .	1146
6.198.3.13nit . . . . .	1146
6.198.3.14sExplicitQosEnabled . . . . .	1146
6.198.3.15sMessageIdEnabled . . . . .	1147
6.198.3.16sMessageTimestampEnabled . . . . .	1147
6.198.3.17sNoLocal . . . . .	1147
6.198.3.18operator= . . . . .	1147
6.198.3.19receive . . . . .	1147
6.198.3.20receive . . . . .	1147
6.198.3.21receive . . . . .	1148
6.198.3.22receiveSelected . . . . .	1148
6.198.3.23receiveSelected . . . . .	1149
6.198.3.24receiveSelected . . . . .	1149
6.198.3.25send . . . . .	1149
6.198.3.26send . . . . .	1150
6.198.3.27send . . . . .	1150
6.198.3.28setDefaultDestination . . . . .	1151
6.198.3.29setDefaultDestinationName . . . . .	1151
6.198.3.30setDeliveryMode . . . . .	1151
6.198.3.31setDeliveryPersistent . . . . .	1151
6.198.3.32setExplicitQosEnabled . . . . .	1152
6.198.3.33setMessageIdEnabled . . . . .	1152
6.198.3.34setMessageTimestampEnabled . . . . .	1152
6.198.3.35setNoLocal . . . . .	1152
6.198.3.36setPriority . . . . .	1152
6.198.3.37setPubSubDomain . . . . .	1152
6.198.3.38setReceiveTimeout . . . . .	1153
6.198.3.39setTimeToLive . . . . .	1153
6.198.4 Friends And Related Function Documentation . . . . .	1153
6.198.4.1 ProducerExecutor . . . . .	1153
6.198.4.2 ReceiveExecutor . . . . .	1153
6.198.4.3 ResolveProducerExecutor . . . . .	1153

6.198.4.4 ResolveReceiveExecutor . . . . .	1153
6.198.4.5 SendExecutor . . . . .	1153
6.198.5 Field Documentation . . . . .	1153
6.198.5.1 DEFAULT_PRIORITY . . . . .	1153
6.198.5.2 DEFAULT_TIME_TO_LIVE . . . . .	1154
6.198.5.3 RECEIVE_TIMEOUT_INDEFINITE_WAIT . . . . .	1154
6.198.5.4 RECEIVE_TIMEOUT_NO_WAIT . . . . .	1154
6.199code Struct Reference . . . . .	1154
6.199.1 Field Documentation . . . . .	1154
6.199.1.1 bits . . . . .	1154
6.199.1.2 op . . . . .	1154
6.199.1.3 val . . . . .	1154
6.200decaf::util::Collection< E > Class Template Reference . . . . .	1155
6.200.1 Detailed Description . . . . .	1156
6.200.2 Constructor & Destructor Documentation . . . . .	1156
6.200.2.1 ~Collection . . . . .	1156
6.200.3 Member Function Documentation . . . . .	1156
6.200.3.1 add . . . . .	1156
6.200.3.2 addAll . . . . .	1158
6.200.3.3 clear . . . . .	1158
6.200.3.4 contains . . . . .	1159
6.200.3.5 containsAll . . . . .	1160
6.200.3.6 equals . . . . .	1161
6.200.3.7 isEmpty . . . . .	1161
6.200.3.8 remove . . . . .	1162
6.200.3.9 removeAll . . . . .	1163
6.200.3.10retainAll . . . . .	1163
6.200.3.11size . . . . .	1164
6.200.3.12toArray . . . . .	1165
6.201activemq::commands::Command Class Reference . . . . .	1165
6.201.1 Constructor & Destructor Documentation . . . . .	1166
6.201.1.1 ~Command . . . . .	1166
6.201.2 Member Function Documentation . . . . .	1166
6.201.2.1 getCommandId . . . . .	1166

6.201.2.2 isBrokerInfo . . . . .	1167
6.201.2.3 isConnectionInfo . . . . .	1167
6.201.2.4 isConsumerInfo . . . . .	1167
6.201.2.5 isKeepAliveInfo . . . . .	1167
6.201.2.6 isMessage . . . . .	1167
6.201.2.7 isMessageAck . . . . .	1167
6.201.2.8 isMessageDispatch . . . . .	1167
6.201.2.9 isMessageDispatchNotification . . . . .	1168
6.201.2.10sProducerAck . . . . .	1168
6.201.2.11sProducerInfo . . . . .	1168
6.201.2.12sRemoveInfo . . . . .	1168
6.201.2.13sRemoveSubscriptionInfo . . . . .	1168
6.201.2.14sResponse . . . . .	1168
6.201.2.15sResponseRequired . . . . .	1168
6.201.2.16sShutdownInfo . . . . .	1169
6.201.2.17sTransactionInfo . . . . .	1169
6.201.2.18sWireFormatInfo . . . . .	1169
6.201.2.19setCommandId . . . . .	1169
6.201.2.20setResponseRequired . . . . .	1169
6.201.2.21toString . . . . .	1169
6.201.2.22visit . . . . .	1170
6.202activemq::state::CommandVisitor Class Reference . . . . .	1171
6.202.1 Detailed Description . . . . .	1173
6.202.2 Constructor & Destructor Documentation . . . . .	1173
6.202.2.1 ~CommandVisitor . . . . .	1173
6.202.3 Member Function Documentation . . . . .	1173
6.202.3.1 processBeginTransaction . . . . .	1173
6.202.3.2 processBrokerError . . . . .	1174
6.202.3.3 processBrokerInfo . . . . .	1174
6.202.3.4 processCommitTransactionOnePhase . . . . .	1174
6.202.3.5 processCommitTransactionTwoPhase . . . . .	1174
6.202.3.6 processConnectionControl . . . . .	1174
6.202.3.7 processConnectionError . . . . .	1174
6.202.3.8 processConnectionInfo . . . . .	1174

6.202.3.9 processConsumerControl . . . . .	1174
6.202.3.10 processConsumerInfo . . . . .	1175
6.202.3.11 processControlCommand . . . . .	1175
6.202.3.12 processDestinationInfo . . . . .	1175
6.202.3.13 processEndTransaction . . . . .	1175
6.202.3.14 processFlushCommand . . . . .	1175
6.202.3.15 processForgetTransaction . . . . .	1175
6.202.3.16 processKeepAliveInfo . . . . .	1175
6.202.3.17 processMessage . . . . .	1175
6.202.3.18 processMessageAck . . . . .	1176
6.202.3.19 processMessageDispatch . . . . .	1176
6.202.3.20 processMessageDispatchNotification . . . . .	1176
6.202.3.21 processMessagePull . . . . .	1176
6.202.3.22 processPrepareTransaction . . . . .	1176
6.202.3.23 processProducerAck . . . . .	1176
6.202.3.24 processProducerInfo . . . . .	1176
6.202.3.25 processRecoverTransactions . . . . .	1176
6.202.3.26 processRemoveConnection . . . . .	1177
6.202.3.27 processRemoveConsumer . . . . .	1177
6.202.3.28 processRemoveDestination . . . . .	1177
6.202.3.29 processRemoveInfo . . . . .	1177
6.202.3.30 processRemoveProducer . . . . .	1177
6.202.3.31 processRemoveSession . . . . .	1177
6.202.3.32 processRemoveSubscriptionInfo . . . . .	1178
6.202.3.33 processReplayCommand . . . . .	1178
6.202.3.34 processResponse . . . . .	1178
6.202.3.35 processRollbackTransaction . . . . .	1178
6.202.3.36 processSessionInfo . . . . .	1178
6.202.3.37 processShutdownInfo . . . . .	1178
6.202.3.38 processTransactionInfo . . . . .	1178
6.202.3.39 processWireFormat . . . . .	1178
6.203 activemq::state::CommandVisitorAdapter Class Reference . . . . .	1179
6.203.1 Detailed Description . . . . .	1181
6.203.2 Constructor & Destructor Documentation . . . . .	1181

6.203.2.1 ~CommandVisitorAdapter . . . . .	1182
6.203.3 Member Function Documentation . . . . .	1182
6.203.3.1 processBeginTransaction . . . . .	1182
6.203.3.2 processBrokerError . . . . .	1182
6.203.3.3 processBrokerInfo . . . . .	1182
6.203.3.4 processCommitTransactionOnePhase . . . . .	1182
6.203.3.5 processCommitTransactionTwoPhase . . . . .	1182
6.203.3.6 processConnectionControl . . . . .	1182
6.203.3.7 processConnectionError . . . . .	1182
6.203.3.8 processConnectionInfo . . . . .	1182
6.203.3.9 processConsumerControl . . . . .	1183
6.203.3.10 processConsumerInfo . . . . .	1183
6.203.3.11 processControlCommand . . . . .	1183
6.203.3.12 processDestinationInfo . . . . .	1183
6.203.3.13 processEndTransaction . . . . .	1183
6.203.3.14 processFlushCommand . . . . .	1183
6.203.3.15 processForgetTransaction . . . . .	1183
6.203.3.16 processKeepAliveInfo . . . . .	1183
6.203.3.17 processMessage . . . . .	1183
6.203.3.18 processMessageAck . . . . .	1184
6.203.3.19 processMessageDispatch . . . . .	1184
6.203.3.20 processMessageDispatchNotification . . . . .	1184
6.203.3.21 processMessagePull . . . . .	1184
6.203.3.22 processPrepareTransaction . . . . .	1184
6.203.3.23 processProducerAck . . . . .	1184
6.203.3.24 processProducerInfo . . . . .	1184
6.203.3.25 processRecoverTransactions . . . . .	1184
6.203.3.26 processRemoveConnection . . . . .	1184
6.203.3.27 processRemoveConsumer . . . . .	1185
6.203.3.28 processRemoveDestination . . . . .	1185
6.203.3.29 processRemoveInfo . . . . .	1185
6.203.3.30 processRemoveProducer . . . . .	1185
6.203.3.31 processRemoveSession . . . . .	1185
6.203.3.32 processRemoveSubscriptionInfo . . . . .	1185

6.203.3.33	processReplayCommand . . . . .	1185
6.203.3.34	processResponse . . . . .	1185
6.203.3.35	processRollbackTransaction . . . . .	1186
6.203.3.36	processSessionInfo . . . . .	1186
6.203.3.37	processShutdownInfo . . . . .	1186
6.203.3.38	processTransactionInfo . . . . .	1186
6.203.3.39	processWireFormat . . . . .	1186
6.204	decaf::lang::Comparable< T > Class Template Reference . . . . .	1186
6.204.1	Detailed Description . . . . .	1187
6.204.2	Constructor & Destructor Documentation . . . . .	1187
6.204.2.1	~Comparable . . . . .	1187
6.204.3	Member Function Documentation . . . . .	1187
6.204.3.1	compareTo . . . . .	1187
6.204.3.2	equals . . . . .	1188
6.204.3.3	operator< . . . . .	1188
6.204.3.4	operator== . . . . .	1189
6.205	decaf::util::Comparator< T > Class Template Reference . . . . .	1189
6.205.1	Detailed Description . . . . .	1189
6.205.2	Constructor & Destructor Documentation . . . . .	1190
6.205.2.1	~Comparator . . . . .	1190
6.205.3	Member Function Documentation . . . . .	1190
6.205.3.1	compare . . . . .	1190
6.205.3.2	operator() . . . . .	1191
6.206	activemq::util::CompositeData Class Reference . . . . .	1191
6.206.1	Detailed Description . . . . .	1192
6.206.2	Constructor & Destructor Documentation . . . . .	1192
6.206.2.1	CompositeData . . . . .	1192
6.206.2.2	~CompositeData . . . . .	1192
6.206.3	Member Function Documentation . . . . .	1192
6.206.3.1	getComponents . . . . .	1192
6.206.3.2	getComponents . . . . .	1192
6.206.3.3	getFragment . . . . .	1192
6.206.3.4	getHost . . . . .	1192
6.206.3.5	getParameters . . . . .	1192



6.206.3.6	getPath . . . . .	1192
6.206.3.7	getScheme . . . . .	1192
6.206.3.8	setComponents . . . . .	1192
6.206.3.9	setFragment . . . . .	1192
6.206.3.10	setHost . . . . .	1192
6.206.3.11	setParameters . . . . .	1192
6.206.3.12	setPath . . . . .	1192
6.206.3.13	setScheme . . . . .	1193
6.206.3.14	toURI . . . . .	1193
6.207	activemq::threads::CompositeTask Class Reference . . . . .	1193
6.207.1	Detailed Description . . . . .	1193
6.207.2	Constructor & Destructor Documentation . . . . .	1193
6.207.2.1	~CompositeTask . . . . .	1193
6.207.3	Member Function Documentation . . . . .	1193
6.207.3.1	isPending . . . . .	1194
6.208	activemq::threads::CompositeTaskRunner Class Reference . . . . .	1194
6.208.1	Detailed Description . . . . .	1195
6.208.2	Constructor & Destructor Documentation . . . . .	1195
6.208.2.1	CompositeTaskRunner . . . . .	1195
6.208.2.2	~CompositeTaskRunner . . . . .	1195
6.208.3	Member Function Documentation . . . . .	1195
6.208.3.1	addTask . . . . .	1195
6.208.3.2	iterate . . . . .	1195
6.208.3.3	removeTask . . . . .	1196
6.208.3.4	run . . . . .	1196
6.208.3.5	shutdown . . . . .	1196
6.208.3.6	shutdown . . . . .	1196
6.208.3.7	wakeup . . . . .	1196
6.209	activemq::transport::CompositeTransport Class Reference . . . . .	1197
6.209.1	Detailed Description . . . . .	1197
6.209.2	Constructor & Destructor Documentation . . . . .	1197
6.209.2.1	~CompositeTransport . . . . .	1197
6.209.3	Member Function Documentation . . . . .	1197
6.209.3.1	addURI . . . . .	1197

6.209.3.2 removeURI . . . . .	1198
6.210decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class	
Template Reference . . . . .	1198
6.210.1 Detailed Description . . . . .	1199
6.210.2 Constructor & Destructor Documentation . . . . .	1199
6.210.2.1 ~ConcurrentMap . . . . .	1199
6.210.3 Member Function Documentation . . . . .	1199
6.210.3.1 putIfAbsent . . . . .	1199
6.210.3.2 remove . . . . .	1200
6.210.3.3 replace . . . . .	1201
6.210.3.4 replace . . . . .	1202
6.211decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class	
Template Reference . . . . .	1203
6.211.1 Detailed Description . . . . .	1206
6.211.2 Constructor & Destructor Documentation . . . . .	1207
6.211.2.1 ConcurrentStlMap . . . . .	1207
6.211.2.2 ConcurrentStlMap . . . . .	1207
6.211.2.3 ConcurrentStlMap . . . . .	1207
6.211.2.4 ~ConcurrentStlMap . . . . .	1207
6.211.3 Member Function Documentation . . . . .	1207
6.211.3.1 clear . . . . .	1208
6.211.3.2 containsKey . . . . .	1208
6.211.3.3 containsValue . . . . .	1208
6.211.3.4 copy . . . . .	1209
6.211.3.5 copy . . . . .	1209
6.211.3.6 equals . . . . .	1209
6.211.3.7 equals . . . . .	1210
6.211.3.8 get . . . . .	1210
6.211.3.9 get . . . . .	1210
6.211.3.10isEmpty . . . . .	1211
6.211.3.11keySet . . . . .	1211
6.211.3.12lock . . . . .	1211
6.211.3.13notify . . . . .	1212
6.211.3.14notifyAll . . . . .	1212

6.211.3.15	put . . . . .	1212
6.211.3.16	putAll . . . . .	1213
6.211.3.17	putAll . . . . .	1213
6.211.3.18	putIfAbsent . . . . .	1214
6.211.3.19	remove . . . . .	1214
6.211.3.20	remove . . . . .	1215
6.211.3.21	replace . . . . .	1215
6.211.3.22	replace . . . . .	1216
6.211.3.23	size . . . . .	1217
6.211.3.24	tryLock . . . . .	1217
6.211.3.25	unlock . . . . .	1217
6.211.3.26	values . . . . .	1218
6.211.3.27	wait . . . . .	1218
6.211.3.28	wait . . . . .	1218
6.211.3.29	wait . . . . .	1219
6.212	decaf::util::concurrent::locks::Condition Class Reference . . . . .	1220
6.212.1	Detailed Description . . . . .	1220
6.212.2	Constructor & Destructor Documentation . . . . .	1222
6.212.2.1	~Condition . . . . .	1222
6.212.3	Member Function Documentation . . . . .	1222
6.212.3.1	await . . . . .	1222
6.212.3.2	await . . . . .	1223
6.212.3.3	awaitNanos . . . . .	1224
6.212.3.4	awaitUninterruptibly . . . . .	1225
6.212.3.5	awaitUntil . . . . .	1226
6.212.3.6	signal . . . . .	1226
6.212.3.7	signalAll . . . . .	1226
6.213	decaf::util::concurrent::ConditionHandle Class Reference . . . . .	1226
6.213.1	Constructor & Destructor Documentation . . . . .	1227
6.213.1.1	ConditionHandle . . . . .	1227
6.213.1.2	~ConditionHandle . . . . .	1227
6.213.1.3	ConditionHandle . . . . .	1227
6.213.1.4	~ConditionHandle . . . . .	1227
6.213.2	Field Documentation . . . . .	1227

6.213.2.1 condition . . . . .	1227
6.213.2.2 criticalSection . . . . .	1227
6.213.2.3 generation . . . . .	1227
6.213.2.4 mutex . . . . .	1227
6.213.2.5 numWaiting . . . . .	1227
6.213.2.6 numWake . . . . .	1227
6.213.2.7 semaphore . . . . .	1227
6.214decaf::internal::util::concurrent::ConditionImpl Class Reference . . . . .	1228
6.214.1 Member Function Documentation . . . . .	1228
6.214.1.1 create . . . . .	1228
6.214.1.2 destroy . . . . .	1228
6.214.1.3 notify . . . . .	1229
6.214.1.4 notifyAll . . . . .	1229
6.214.1.5 wait . . . . .	1229
6.214.1.6 wait . . . . .	1229
6.215decaf::net::ConnectException Class Reference . . . . .	1230
6.215.1 Constructor & Destructor Documentation . . . . .	1230
6.215.1.1 ConnectException . . . . .	1230
6.215.1.2 ConnectException . . . . .	1230
6.215.1.3 ConnectException . . . . .	1231
6.215.1.4 ConnectException . . . . .	1231
6.215.1.5 ConnectException . . . . .	1231
6.215.1.6 ConnectException . . . . .	1231
6.215.1.7 ~ConnectException . . . . .	1232
6.215.2 Member Function Documentation . . . . .	1232
6.215.2.1 clone . . . . .	1232
6.216cms::Connection Class Reference . . . . .	1232
6.216.1 Detailed Description . . . . .	1233
6.216.2 Constructor & Destructor Documentation . . . . .	1234
6.216.2.1 ~Connection . . . . .	1234
6.216.3 Member Function Documentation . . . . .	1234
6.216.3.1 close . . . . .	1234
6.216.3.2 createSession . . . . .	1234
6.216.3.3 createSession . . . . .	1235

6.216.3.4	getClientID . . . . .	1235
6.216.3.5	getExceptionListener . . . . .	1235
6.216.3.6	getMetaData . . . . .	1235
6.216.3.7	setClientID . . . . .	1236
6.216.3.8	setExceptionListener . . . . .	1236
6.217	activemq::commands::ConnectionControl Class Reference . . . . .	1237
6.217.1	Constructor & Destructor Documentation . . . . .	1238
6.217.1.1	ConnectionControl . . . . .	1238
6.217.1.2	~ConnectionControl . . . . .	1238
6.217.2	Member Function Documentation . . . . .	1238
6.217.2.1	cloneDataStructure . . . . .	1238
6.217.2.2	copyDataStructure . . . . .	1239
6.217.2.3	equals . . . . .	1239
6.217.2.4	getConnectedBrokers . . . . .	1239
6.217.2.5	getConnectedBrokers . . . . .	1239
6.217.2.6	getDataStructureType . . . . .	1239
6.217.2.7	getReconnectTo . . . . .	1239
6.217.2.8	getReconnectTo . . . . .	1240
6.217.2.9	isClose . . . . .	1240
6.217.2.10	isExit . . . . .	1240
6.217.2.11	isFaultTolerant . . . . .	1240
6.217.2.12	isRebalanceConnection . . . . .	1240
6.217.2.13	isResume . . . . .	1240
6.217.2.14	isSuspend . . . . .	1240
6.217.2.15	setClose . . . . .	1240
6.217.2.16	setConnectedBrokers . . . . .	1240
6.217.2.17	setExit . . . . .	1240
6.217.2.18	setFaultTolerant . . . . .	1240
6.217.2.19	setRebalanceConnection . . . . .	1240
6.217.2.20	setReconnectTo . . . . .	1240
6.217.2.21	setResume . . . . .	1240
6.217.2.22	setSuspend . . . . .	1240
6.217.2.23	toString . . . . .	1241
6.217.2.24	visit . . . . .	1241

6.217.3 Field Documentation . . . . .	1241
6.217.3.1 close . . . . .	1241
6.217.3.2 connectedBrokers . . . . .	1241
6.217.3.3 exit . . . . .	1241
6.217.3.4 faultTolerant . . . . .	1241
6.217.3.5 ID_CONNECTIONCONTROL . . . . .	1241
6.217.3.6 rebalanceConnection . . . . .	1241
6.217.3.7 reconnectTo . . . . .	1241
6.217.3.8 resume . . . . .	1242
6.217.3.9 suspend . . . . .	1242
6.218activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	
Class Reference . . . . .	1242
6.218.1 Detailed Description . . . . .	1243
6.218.2 Constructor & Destructor Documentation . . . . .	1243
6.218.2.1 ConnectionControlMarshaller . . . . .	1243
6.218.2.2 ~ConnectionControlMarshaller . . . . .	1243
6.218.3 Member Function Documentation . . . . .	1243
6.218.3.1 createObject . . . . .	1243
6.218.3.2 getDataStructureType . . . . .	1243
6.218.3.3 looseMarshal . . . . .	1244
6.218.3.4 looseUnmarshal . . . . .	1244
6.218.3.5 tightMarshal1 . . . . .	1244
6.218.3.6 tightMarshal2 . . . . .	1245
6.218.3.7 tightUnmarshal . . . . .	1245
6.219activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	
Class Reference . . . . .	1246
6.219.1 Detailed Description . . . . .	1247
6.219.2 Constructor & Destructor Documentation . . . . .	1247
6.219.2.1 ConnectionControlMarshaller . . . . .	1247
6.219.2.2 ~ConnectionControlMarshaller . . . . .	1247
6.219.3 Member Function Documentation . . . . .	1247
6.219.3.1 createObject . . . . .	1247
6.219.3.2 getDataStructureType . . . . .	1247
6.219.3.3 looseMarshal . . . . .	1248

6.219.3.4 looseUnmarshal . . . . .	1248
6.219.3.5 tightMarshal1 . . . . .	1248
6.219.3.6 tightMarshal2 . . . . .	1249
6.219.3.7 tightUnmarshal . . . . .	1249
6.220activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	
Class Reference . . . . .	1250
6.220.1 Detailed Description . . . . .	1251
6.220.2 Constructor & Destructor Documentation . . . . .	1251
6.220.2.1 ConnectionControlMarshaller . . . . .	1251
6.220.2.2 ~ConnectionControlMarshaller . . . . .	1251
6.220.3 Member Function Documentation . . . . .	1251
6.220.3.1 createObject . . . . .	1251
6.220.3.2 getDataStructureType . . . . .	1251
6.220.3.3 looseMarshal . . . . .	1252
6.220.3.4 looseUnmarshal . . . . .	1252
6.220.3.5 tightMarshal1 . . . . .	1252
6.220.3.6 tightMarshal2 . . . . .	1253
6.220.3.7 tightUnmarshal . . . . .	1253
6.221activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	
Class Reference . . . . .	1254
6.221.1 Detailed Description . . . . .	1255
6.221.2 Constructor & Destructor Documentation . . . . .	1255
6.221.2.1 ConnectionControlMarshaller . . . . .	1255
6.221.2.2 ~ConnectionControlMarshaller . . . . .	1255
6.221.3 Member Function Documentation . . . . .	1255
6.221.3.1 createObject . . . . .	1255
6.221.3.2 getDataStructureType . . . . .	1255
6.221.3.3 looseMarshal . . . . .	1256
6.221.3.4 looseUnmarshal . . . . .	1256
6.221.3.5 tightMarshal1 . . . . .	1256
6.221.3.6 tightMarshal2 . . . . .	1257
6.221.3.7 tightUnmarshal . . . . .	1257
6.222activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller	
Class Reference . . . . .	1258
6.222.1 Detailed Description . . . . .	1259

6.222.2 Constructor & Destructor Documentation . . . . .	1259
6.222.2.1 ConnectionControlMarshaller . . . . .	1259
6.222.2.2 ~ConnectionControlMarshaller . . . . .	1259
6.222.3 Member Function Documentation . . . . .	1259
6.222.3.1 createObject . . . . .	1259
6.222.3.2 getDataStructureType . . . . .	1259
6.222.3.3 looseMarshal . . . . .	1260
6.222.3.4 looseUnmarshal . . . . .	1260
6.222.3.5 tightMarshal1 . . . . .	1260
6.222.3.6 tightMarshal2 . . . . .	1261
6.222.3.7 tightUnmarshal . . . . .	1261
6.223activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	
Class Reference . . . . .	1262
6.223.1 Detailed Description . . . . .	1263
6.223.2 Constructor & Destructor Documentation . . . . .	1263
6.223.2.1 ConnectionControlMarshaller . . . . .	1263
6.223.2.2 ~ConnectionControlMarshaller . . . . .	1263
6.223.3 Member Function Documentation . . . . .	1263
6.223.3.1 createObject . . . . .	1263
6.223.3.2 getDataStructureType . . . . .	1263
6.223.3.3 looseMarshal . . . . .	1264
6.223.3.4 looseUnmarshal . . . . .	1264
6.223.3.5 tightMarshal1 . . . . .	1264
6.223.3.6 tightMarshal2 . . . . .	1265
6.223.3.7 tightUnmarshal . . . . .	1265
6.224activemq::commands::ConnectionError Class Reference . . . . .	1266
6.224.1 Constructor & Destructor Documentation . . . . .	1267
6.224.1.1 ConnectionError . . . . .	1267
6.224.1.2 ~ConnectionError . . . . .	1267
6.224.2 Member Function Documentation . . . . .	1267
6.224.2.1 cloneDataStructure . . . . .	1267
6.224.2.2 copyDataStructure . . . . .	1267
6.224.2.3 equals . . . . .	1268
6.224.2.4 getConnectionId . . . . .	1268



6.224.2.5	getConnectionId . . . . .	1268
6.224.2.6	getDataStructureType . . . . .	1268
6.224.2.7	getException . . . . .	1268
6.224.2.8	getException . . . . .	1268
6.224.2.9	setConnectionId . . . . .	1268
6.224.2.10	setException . . . . .	1269
6.224.2.11	toString . . . . .	1269
6.224.2.12	visit . . . . .	1269
6.224.3	Field Documentation . . . . .	1269
6.224.3.1	connectionId . . . . .	1269
6.224.3.2	exception . . . . .	1269
6.224.3.3	ID_CONNECTIONERROR . . . . .	1269
6.225	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	
	Class Reference . . . . .	1270
6.225.1	Detailed Description . . . . .	1270
6.225.2	Constructor & Destructor Documentation . . . . .	1271
6.225.2.1	ConnectionErrorMarshaller . . . . .	1271
6.225.2.2	~ConnectionErrorMarshaller . . . . .	1271
6.225.3	Member Function Documentation . . . . .	1271
6.225.3.1	createObject . . . . .	1271
6.225.3.2	getDataStructureType . . . . .	1271
6.225.3.3	looseMarshal . . . . .	1271
6.225.3.4	looseUnmarshal . . . . .	1272
6.225.3.5	tightMarshal1 . . . . .	1272
6.225.3.6	tightMarshal2 . . . . .	1273
6.225.3.7	tightUnmarshal . . . . .	1273
6.226	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	
	Class Reference . . . . .	1274
6.226.1	Detailed Description . . . . .	1274
6.226.2	Constructor & Destructor Documentation . . . . .	1275
6.226.2.1	ConnectionErrorMarshaller . . . . .	1275
6.226.2.2	~ConnectionErrorMarshaller . . . . .	1275
6.226.3	Member Function Documentation . . . . .	1275
6.226.3.1	createObject . . . . .	1275

6.226.3.2	getDataStructureType . . . . .	1275
6.226.3.3	looseMarshal . . . . .	1275
6.226.3.4	looseUnmarshal . . . . .	1276
6.226.3.5	tightMarshal1 . . . . .	1276
6.226.3.6	tightMarshal2 . . . . .	1277
6.226.3.7	tightUnmarshal . . . . .	1277
6.227	activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	
	Class Reference . . . . .	1278
6.227.1	Detailed Description . . . . .	1278
6.227.2	Constructor & Destructor Documentation . . . . .	1279
6.227.2.1	ConnectionErrorMarshaller . . . . .	1279
6.227.2.2	~ConnectionErrorMarshaller . . . . .	1279
6.227.3	Member Function Documentation . . . . .	1279
6.227.3.1	createObject . . . . .	1279
6.227.3.2	getDataStructureType . . . . .	1279
6.227.3.3	looseMarshal . . . . .	1279
6.227.3.4	looseUnmarshal . . . . .	1280
6.227.3.5	tightMarshal1 . . . . .	1280
6.227.3.6	tightMarshal2 . . . . .	1281
6.227.3.7	tightUnmarshal . . . . .	1281
6.228	activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	
	Class Reference . . . . .	1282
6.228.1	Detailed Description . . . . .	1282
6.228.2	Constructor & Destructor Documentation . . . . .	1283
6.228.2.1	ConnectionErrorMarshaller . . . . .	1283
6.228.2.2	~ConnectionErrorMarshaller . . . . .	1283
6.228.3	Member Function Documentation . . . . .	1283
6.228.3.1	createObject . . . . .	1283
6.228.3.2	getDataStructureType . . . . .	1283
6.228.3.3	looseMarshal . . . . .	1283
6.228.3.4	looseUnmarshal . . . . .	1284
6.228.3.5	tightMarshal1 . . . . .	1284
6.228.3.6	tightMarshal2 . . . . .	1285
6.228.3.7	tightUnmarshal . . . . .	1285

6.229activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	
Class Reference . . . . .	1286
6.229.1 Detailed Description . . . . .	1286
6.229.2 Constructor & Destructor Documentation . . . . .	1287
6.229.2.1 ConnectionErrorMarshaller . . . . .	1287
6.229.2.2 ~ConnectionErrorMarshaller . . . . .	1287
6.229.3 Member Function Documentation . . . . .	1287
6.229.3.1 createObject . . . . .	1287
6.229.3.2 getDataStructureType . . . . .	1287
6.229.3.3 looseMarshal . . . . .	1287
6.229.3.4 looseUnmarshal . . . . .	1288
6.229.3.5 tightMarshal1 . . . . .	1288
6.229.3.6 tightMarshal2 . . . . .	1289
6.229.3.7 tightUnmarshal . . . . .	1289
6.230activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller	
Class Reference . . . . .	1290
6.230.1 Detailed Description . . . . .	1290
6.230.2 Constructor & Destructor Documentation . . . . .	1291
6.230.2.1 ConnectionErrorMarshaller . . . . .	1291
6.230.2.2 ~ConnectionErrorMarshaller . . . . .	1291
6.230.3 Member Function Documentation . . . . .	1291
6.230.3.1 createObject . . . . .	1291
6.230.3.2 getDataStructureType . . . . .	1291
6.230.3.3 looseMarshal . . . . .	1291
6.230.3.4 looseUnmarshal . . . . .	1292
6.230.3.5 tightMarshal1 . . . . .	1292
6.230.3.6 tightMarshal2 . . . . .	1293
6.230.3.7 tightUnmarshal . . . . .	1293
6.231cms::ConnectionFactory Class Reference . . . . .	1294
6.231.1 Detailed Description . . . . .	1294
6.231.2 Constructor & Destructor Documentation . . . . .	1295
6.231.2.1 ~ConnectionFactory . . . . .	1295
6.231.3 Member Function Documentation . . . . .	1295
6.231.3.1 createCMSConnectionFactory . . . . .	1295

6.231.3.2 createConnection . . . . .	1295
6.231.3.3 createConnection . . . . .	1296
6.231.3.4 createConnection . . . . .	1296
6.232activemq::commands::ConnectionId Class Reference . . . . .	1297
6.232.1 Member Typedef Documentation . . . . .	1298
6.232.1.1 COMPARATOR . . . . .	1298
6.232.2 Constructor & Destructor Documentation . . . . .	1298
6.232.2.1 ConnectionId . . . . .	1298
6.232.2.2 ConnectionId . . . . .	1298
6.232.2.3 ConnectionId . . . . .	1298
6.232.2.4 ConnectionId . . . . .	1298
6.232.2.5 ConnectionId . . . . .	1298
6.232.2.6 ~ConnectionId . . . . .	1298
6.232.3 Member Function Documentation . . . . .	1298
6.232.3.1 cloneDataStructure . . . . .	1298
6.232.3.2 compareTo . . . . .	1299
6.232.3.3 copyDataStructure . . . . .	1299
6.232.3.4 equals . . . . .	1299
6.232.3.5 equals . . . . .	1299
6.232.3.6 getDataStructureType . . . . .	1299
6.232.3.7 getValue . . . . .	1300
6.232.3.8 getValue . . . . .	1300
6.232.3.9 operator< . . . . .	1300
6.232.3.10operator= . . . . .	1300
6.232.3.11operator== . . . . .	1300
6.232.3.12setValue . . . . .	1300
6.232.3.13toString . . . . .	1300
6.232.4 Field Documentation . . . . .	1300
6.232.4.1 ID_CONNECTIONID . . . . .	1300
6.232.4.2 value . . . . .	1300
6.233activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Reference . . . . .	1301
6.233.1 Detailed Description . . . . .	1301
6.233.2 Constructor & Destructor Documentation . . . . .	1302

6.233.2.1 ConnectionIdMarshaller . . . . .	1302
6.233.2.2 ~ConnectionIdMarshaller . . . . .	1302
6.233.3 Member Function Documentation . . . . .	1302
6.233.3.1 createObject . . . . .	1302
6.233.3.2 getDataStructureType . . . . .	1302
6.233.3.3 looseMarshal . . . . .	1302
6.233.3.4 looseUnmarshal . . . . .	1303
6.233.3.5 tightMarshal1 . . . . .	1303
6.233.3.6 tightMarshal2 . . . . .	1304
6.233.3.7 tightUnmarshal . . . . .	1304
6.234activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class	
Reference . . . . .	1305
6.234.1 Detailed Description . . . . .	1305
6.234.2 Constructor & Destructor Documentation . . . . .	1306
6.234.2.1 ConnectionIdMarshaller . . . . .	1306
6.234.2.2 ~ConnectionIdMarshaller . . . . .	1306
6.234.3 Member Function Documentation . . . . .	1306
6.234.3.1 createObject . . . . .	1306
6.234.3.2 getDataStructureType . . . . .	1306
6.234.3.3 looseMarshal . . . . .	1306
6.234.3.4 looseUnmarshal . . . . .	1307
6.234.3.5 tightMarshal1 . . . . .	1307
6.234.3.6 tightMarshal2 . . . . .	1308
6.234.3.7 tightUnmarshal . . . . .	1308
6.235activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class	
Reference . . . . .	1309
6.235.1 Detailed Description . . . . .	1309
6.235.2 Constructor & Destructor Documentation . . . . .	1310
6.235.2.1 ConnectionIdMarshaller . . . . .	1310
6.235.2.2 ~ConnectionIdMarshaller . . . . .	1310
6.235.3 Member Function Documentation . . . . .	1310
6.235.3.1 createObject . . . . .	1310
6.235.3.2 getDataStructureType . . . . .	1310
6.235.3.3 looseMarshal . . . . .	1310

6.235.3.4 looseUnmarshal . . . . .	1311
6.235.3.5 tightMarshal1 . . . . .	1311
6.235.3.6 tightMarshal2 . . . . .	1312
6.235.3.7 tightUnmarshal . . . . .	1312
6.236activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference . . . . .	1313
6.236.1 Detailed Description . . . . .	1313
6.236.2 Constructor & Destructor Documentation . . . . .	1314
6.236.2.1 ConnectionIdMarshaller . . . . .	1314
6.236.2.2 ~ConnectionIdMarshaller . . . . .	1314
6.236.3 Member Function Documentation . . . . .	1314
6.236.3.1 createObject . . . . .	1314
6.236.3.2 getDataStructureType . . . . .	1314
6.236.3.3 looseMarshal . . . . .	1314
6.236.3.4 looseUnmarshal . . . . .	1315
6.236.3.5 tightMarshal1 . . . . .	1315
6.236.3.6 tightMarshal2 . . . . .	1316
6.236.3.7 tightUnmarshal . . . . .	1316
6.237activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference . . . . .	1317
6.237.1 Detailed Description . . . . .	1317
6.237.2 Constructor & Destructor Documentation . . . . .	1318
6.237.2.1 ConnectionIdMarshaller . . . . .	1318
6.237.2.2 ~ConnectionIdMarshaller . . . . .	1318
6.237.3 Member Function Documentation . . . . .	1318
6.237.3.1 createObject . . . . .	1318
6.237.3.2 getDataStructureType . . . . .	1318
6.237.3.3 looseMarshal . . . . .	1318
6.237.3.4 looseUnmarshal . . . . .	1319
6.237.3.5 tightMarshal1 . . . . .	1319
6.237.3.6 tightMarshal2 . . . . .	1320
6.237.3.7 tightUnmarshal . . . . .	1320
6.238activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller Class Reference . . . . .	1321
6.238.1 Detailed Description . . . . .	1321

6.238.2 Constructor & Destructor Documentation . . . . .	1322
6.238.2.1 ConnectionIdMarshaller . . . . .	1322
6.238.2.2 ~ConnectionIdMarshaller . . . . .	1322
6.238.3 Member Function Documentation . . . . .	1322
6.238.3.1 createObject . . . . .	1322
6.238.3.2 getDataStructureType . . . . .	1322
6.238.3.3 looseMarshal . . . . .	1322
6.238.3.4 looseUnmarshal . . . . .	1323
6.238.3.5 tightMarshal1 . . . . .	1323
6.238.3.6 tightMarshal2 . . . . .	1324
6.238.3.7 tightUnmarshal . . . . .	1324
6.239activemq::commands::ConnectionInfo Class Reference . . . . .	1324
6.239.1 Constructor & Destructor Documentation . . . . .	1326
6.239.1.1 ConnectionInfo . . . . .	1326
6.239.1.2 ~ConnectionInfo . . . . .	1326
6.239.2 Member Function Documentation . . . . .	1326
6.239.2.1 cloneDataStructure . . . . .	1326
6.239.2.2 copyDataStructure . . . . .	1327
6.239.2.3 createRemoveCommand . . . . .	1327
6.239.2.4 equals . . . . .	1327
6.239.2.5 getBrokerPath . . . . .	1327
6.239.2.6 getBrokerPath . . . . .	1327
6.239.2.7 getClientId . . . . .	1327
6.239.2.8 getClientId . . . . .	1327
6.239.2.9 getConnectionId . . . . .	1327
6.239.2.10getConnectionId . . . . .	1327
6.239.2.11getDataStructureType . . . . .	1328
6.239.2.12getPassword . . . . .	1328
6.239.2.13getPassword . . . . .	1328
6.239.2.14getUserName . . . . .	1328
6.239.2.15getUserName . . . . .	1328
6.239.2.16sBrokerMasterConnector . . . . .	1328
6.239.2.17sClientMaster . . . . .	1328
6.239.2.18sConnectionInfo . . . . .	1328

6.239.2.19	setFaultTolerant . . . . .	1328
6.239.2.20	setManageable . . . . .	1328
6.239.2.21	setBrokerMasterConnector . . . . .	1328
6.239.2.22	setBrokerPath . . . . .	1329
6.239.2.23	setClientId . . . . .	1329
6.239.2.24	setClientMaster . . . . .	1329
6.239.2.25	setConnectionId . . . . .	1329
6.239.2.26	setFaultTolerant . . . . .	1329
6.239.2.27	setManageable . . . . .	1329
6.239.2.28	setPassword . . . . .	1329
6.239.2.29	setUserName . . . . .	1329
6.239.2.30	toString . . . . .	1329
6.239.2.31	visit . . . . .	1329
6.239.3	Field Documentation . . . . .	1330
6.239.3.1	brokerMasterConnector . . . . .	1330
6.239.3.2	brokerPath . . . . .	1330
6.239.3.3	clientId . . . . .	1330
6.239.3.4	clientMaster . . . . .	1330
6.239.3.5	connectionId . . . . .	1330
6.239.3.6	faultTolerant . . . . .	1330
6.239.3.7	ID_CONNECTIONINFO . . . . .	1330
6.239.3.8	manageable . . . . .	1330
6.239.3.9	password . . . . .	1330
6.239.3.10	userName . . . . .	1330
6.240	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference . . . . .	1330
6.240.1	Detailed Description . . . . .	1331
6.240.2	Constructor & Destructor Documentation . . . . .	1331
6.240.2.1	ConnectionInfoMarshaller . . . . .	1331
6.240.2.2	~ConnectionInfoMarshaller . . . . .	1331
6.240.3	Member Function Documentation . . . . .	1331
6.240.3.1	createObject . . . . .	1332
6.240.3.2	getDataStructureType . . . . .	1332
6.240.3.3	looseMarshal . . . . .	1332



6.240.3.4 looseUnmarshal . . . . .	1333
6.240.3.5 tightMarshal1 . . . . .	1333
6.240.3.6 tightMarshal2 . . . . .	1334
6.240.3.7 tightUnmarshal . . . . .	1334
6.241activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference . . . . .	1335
6.241.1 Detailed Description . . . . .	1335
6.241.2 Constructor & Destructor Documentation . . . . .	1336
6.241.2.1 ConnectionInfoMarshaller . . . . .	1336
6.241.2.2 ~ConnectionInfoMarshaller . . . . .	1336
6.241.3 Member Function Documentation . . . . .	1336
6.241.3.1 createObject . . . . .	1336
6.241.3.2 getDataStructureType . . . . .	1336
6.241.3.3 looseMarshal . . . . .	1336
6.241.3.4 looseUnmarshal . . . . .	1337
6.241.3.5 tightMarshal1 . . . . .	1337
6.241.3.6 tightMarshal2 . . . . .	1338
6.241.3.7 tightUnmarshal . . . . .	1338
6.242activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference . . . . .	1339
6.242.1 Detailed Description . . . . .	1339
6.242.2 Constructor & Destructor Documentation . . . . .	1340
6.242.2.1 ConnectionInfoMarshaller . . . . .	1340
6.242.2.2 ~ConnectionInfoMarshaller . . . . .	1340
6.242.3 Member Function Documentation . . . . .	1340
6.242.3.1 createObject . . . . .	1340
6.242.3.2 getDataStructureType . . . . .	1340
6.242.3.3 looseMarshal . . . . .	1340
6.242.3.4 looseUnmarshal . . . . .	1341
6.242.3.5 tightMarshal1 . . . . .	1341
6.242.3.6 tightMarshal2 . . . . .	1342
6.242.3.7 tightUnmarshal . . . . .	1342
6.243activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference . . . . .	1343
6.243.1 Detailed Description . . . . .	1343

6.243.2 Constructor & Destructor Documentation . . . . .	1344
6.243.2.1 ConnectionInfoMarshaller . . . . .	1344
6.243.2.2 ~ConnectionInfoMarshaller . . . . .	1344
6.243.3 Member Function Documentation . . . . .	1344
6.243.3.1 createObject . . . . .	1344
6.243.3.2 getDataStructureType . . . . .	1344
6.243.3.3 looseMarshal . . . . .	1344
6.243.3.4 looseUnmarshal . . . . .	1345
6.243.3.5 tightMarshal1 . . . . .	1345
6.243.3.6 tightMarshal2 . . . . .	1346
6.243.3.7 tightUnmarshal . . . . .	1346
6.244activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference . . . . .	1347
6.244.1 Detailed Description . . . . .	1347
6.244.2 Constructor & Destructor Documentation . . . . .	1348
6.244.2.1 ConnectionInfoMarshaller . . . . .	1348
6.244.2.2 ~ConnectionInfoMarshaller . . . . .	1348
6.244.3 Member Function Documentation . . . . .	1348
6.244.3.1 createObject . . . . .	1348
6.244.3.2 getDataStructureType . . . . .	1348
6.244.3.3 looseMarshal . . . . .	1348
6.244.3.4 looseUnmarshal . . . . .	1349
6.244.3.5 tightMarshal1 . . . . .	1349
6.244.3.6 tightMarshal2 . . . . .	1350
6.244.3.7 tightUnmarshal . . . . .	1350
6.245activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller Class Reference . . . . .	1351
6.245.1 Detailed Description . . . . .	1351
6.245.2 Constructor & Destructor Documentation . . . . .	1352
6.245.2.1 ConnectionInfoMarshaller . . . . .	1352
6.245.2.2 ~ConnectionInfoMarshaller . . . . .	1352
6.245.3 Member Function Documentation . . . . .	1352
6.245.3.1 createObject . . . . .	1352
6.245.3.2 getDataStructureType . . . . .	1352

6.245.3.3 looseMarshal . . . . .	1352
6.245.3.4 looseUnmarshal . . . . .	1353
6.245.3.5 tightMarshal1 . . . . .	1353
6.245.3.6 tightMarshal2 . . . . .	1354
6.245.3.7 tightUnmarshal . . . . .	1354
6.246cms::ConnectionMetaData Class Reference . . . . .	1355
6.246.1 Detailed Description . . . . .	1355
6.246.2 Constructor & Destructor Documentation . . . . .	1356
6.246.2.1 ~ConnectionMetaData . . . . .	1356
6.246.3 Member Function Documentation . . . . .	1356
6.246.3.1 getCMSMajorVersion . . . . .	1356
6.246.3.2 getCMSMinorVersion . . . . .	1356
6.246.3.3 getCMSProviderName . . . . .	1356
6.246.3.4 getCMSVersion . . . . .	1357
6.246.3.5 getCMSXPropertyNames . . . . .	1357
6.246.3.6 getProviderMajorVersion . . . . .	1357
6.246.3.7 getProviderMinorVersion . . . . .	1358
6.246.3.8 getProviderVersion . . . . .	1358
6.247activemq::state::ConnectionState Class Reference . . . . .	1358
6.247.1 Constructor & Destructor Documentation . . . . .	1359
6.247.1.1 ConnectionState . . . . .	1359
6.247.1.2 ~ConnectionState . . . . .	1359
6.247.2 Member Function Documentation . . . . .	1359
6.247.2.1 addSession . . . . .	1359
6.247.2.2 addTempDestination . . . . .	1359
6.247.2.3 addTransactionState . . . . .	1359
6.247.2.4 checkShutdown . . . . .	1360
6.247.2.5 getInfo . . . . .	1360
6.247.2.6 getRecoveringPullConsumers . . . . .	1360
6.247.2.7 getSessionState . . . . .	1360
6.247.2.8 getSessionStates . . . . .	1360
6.247.2.9 getTempDesinations . . . . .	1360
6.247.2.10getTransactionState . . . . .	1360
6.247.2.11getTransactionStates . . . . .	1360

6.247.2.12	sConnectionInterruptProcessingComplete . . . . .	1360
6.247.2.13	removeSession . . . . .	1360
6.247.2.14	removeTempDestination . . . . .	1360
6.247.2.15	removeTransactionState . . . . .	1360
6.247.2.16	reset . . . . .	1360
6.247.2.17	setConnectionInterruptProcessingComplete . . . . .	1361
6.247.2.18	shutdown . . . . .	1361
6.247.2.19	toString . . . . .	1361
6.248	activemq::state::ConnectionStateTracker Class Reference . . . . .	1361
6.248.1	Constructor & Destructor Documentation . . . . .	1362
6.248.1.1	ConnectionStateTracker . . . . .	1362
6.248.1.2	~ConnectionStateTracker . . . . .	1363
6.248.2	Member Function Documentation . . . . .	1363
6.248.2.1	connectionInterruptProcessingComplete . . . . .	1363
6.248.2.2	getMaxCacheSize . . . . .	1363
6.248.2.3	isRestoreConsumers . . . . .	1363
6.248.2.4	isRestoreProducers . . . . .	1363
6.248.2.5	isRestoreSessions . . . . .	1363
6.248.2.6	isRestoreTransaction . . . . .	1363
6.248.2.7	isTrackMessages . . . . .	1363
6.248.2.8	isTrackTransactionProducers . . . . .	1363
6.248.2.9	isTrackTransactions . . . . .	1363
6.248.2.10	processBeginTransaction . . . . .	1363
6.248.2.11	processCommitTransactionOnePhase . . . . .	1363
6.248.2.12	processCommitTransactionTwoPhase . . . . .	1364
6.248.2.13	processConnectionInfo . . . . .	1364
6.248.2.14	processConsumerInfo . . . . .	1364
6.248.2.15	processDestinationInfo . . . . .	1364
6.248.2.16	processEndTransaction . . . . .	1364
6.248.2.17	processMessage . . . . .	1364
6.248.2.18	processMessageAck . . . . .	1365
6.248.2.19	processPrepareTransaction . . . . .	1365
6.248.2.20	processProducerInfo . . . . .	1365
6.248.2.21	processRemoveConnection . . . . .	1365

6.248.2.22	processRemoveConsumer . . . . .	1365
6.248.2.23	processRemoveDestination . . . . .	1365
6.248.2.24	processRemoveProducer . . . . .	1366
6.248.2.25	processRemoveSession . . . . .	1366
6.248.2.26	processRollbackTransaction . . . . .	1366
6.248.2.27	processSessionInfo . . . . .	1366
6.248.2.28	restore . . . . .	1366
6.248.2.29	setMaxCacheSize . . . . .	1366
6.248.2.30	setRestoreConsumers . . . . .	1366
6.248.2.31	setRestoreProducers . . . . .	1366
6.248.2.32	setRestoreSessions . . . . .	1366
6.248.2.33	setRestoreTransaction . . . . .	1366
6.248.2.34	setTrackMessages . . . . .	1367
6.248.2.35	setTrackTransactionProducers . . . . .	1367
6.248.2.36	setTrackTransactions . . . . .	1367
6.248.2.37	track . . . . .	1367
6.248.2.38	trackBack . . . . .	1367
6.248.2.39	transportInterrupted . . . . .	1367
6.248.3	Friends And Related Function Documentation . . . . .	1367
6.248.3.1	RemoveTransactionAction . . . . .	1367
6.249	decaf::util::logging::ConsoleHandler Class Reference . . . . .	1367
6.249.1	Detailed Description . . . . .	1368
6.249.2	Constructor & Destructor Documentation . . . . .	1368
6.249.2.1	ConsoleHandler . . . . .	1368
6.249.2.2	~ConsoleHandler . . . . .	1368
6.249.3	Member Function Documentation . . . . .	1368
6.249.3.1	close . . . . .	1368
6.249.3.2	publish . . . . .	1368
6.250	activemq::commands::ConsumerControl Class Reference . . . . .	1369
6.250.1	Constructor & Destructor Documentation . . . . .	1370
6.250.1.1	ConsumerControl . . . . .	1370
6.250.1.2	~ConsumerControl . . . . .	1370
6.250.2	Member Function Documentation . . . . .	1370
6.250.2.1	cloneDataStructure . . . . .	1370

6.250.2.2 copyDataStructure . . . . .	1371
6.250.2.3 equals . . . . .	1371
6.250.2.4 getConsumerId . . . . .	1371
6.250.2.5 getConsumerId . . . . .	1371
6.250.2.6 getDataStructureType . . . . .	1371
6.250.2.7 getDestination . . . . .	1371
6.250.2.8 getDestination . . . . .	1372
6.250.2.9 getPrefetch . . . . .	1372
6.250.2.10sClose . . . . .	1372
6.250.2.11isFlush . . . . .	1372
6.250.2.12sStart . . . . .	1372
6.250.2.13sStop . . . . .	1372
6.250.2.14setClose . . . . .	1372
6.250.2.15setConsumerId . . . . .	1372
6.250.2.16setDestination . . . . .	1372
6.250.2.17setFlush . . . . .	1372
6.250.2.18setPrefetch . . . . .	1372
6.250.2.19setStart . . . . .	1372
6.250.2.20setStop . . . . .	1372
6.250.2.21toString . . . . .	1372
6.250.2.22visit . . . . .	1373
6.250.3 Field Documentation . . . . .	1373
6.250.3.1 close . . . . .	1373
6.250.3.2 consumerId . . . . .	1373
6.250.3.3 destination . . . . .	1373
6.250.3.4 flush . . . . .	1373
6.250.3.5 ID_CONSUMERCONTROL . . . . .	1373
6.250.3.6 prefetch . . . . .	1373
6.250.3.7 start . . . . .	1373
6.250.3.8 stop . . . . .	1373
6.251 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller	
Class Reference . . . . .	1373
6.251.1 Detailed Description . . . . .	1374
6.251.2 Constructor & Destructor Documentation . . . . .	1374

6.251.2.1 ConsumerControlMarshaller . . . . .	1374
6.251.2.2 ~ConsumerControlMarshaller . . . . .	1375
6.251.3 Member Function Documentation . . . . .	1375
6.251.3.1 createObject . . . . .	1375
6.251.3.2 getDataStructureType . . . . .	1375
6.251.3.3 looseMarshal . . . . .	1375
6.251.3.4 looseUnmarshal . . . . .	1376
6.251.3.5 tightMarshal1 . . . . .	1376
6.251.3.6 tightMarshal2 . . . . .	1377
6.251.3.7 tightUnmarshal . . . . .	1377
6.252activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	
Class Reference . . . . .	1378
6.252.1 Detailed Description . . . . .	1378
6.252.2 Constructor & Destructor Documentation . . . . .	1379
6.252.2.1 ConsumerControlMarshaller . . . . .	1379
6.252.2.2 ~ConsumerControlMarshaller . . . . .	1379
6.252.3 Member Function Documentation . . . . .	1379
6.252.3.1 createObject . . . . .	1379
6.252.3.2 getDataStructureType . . . . .	1379
6.252.3.3 looseMarshal . . . . .	1379
6.252.3.4 looseUnmarshal . . . . .	1380
6.252.3.5 tightMarshal1 . . . . .	1380
6.252.3.6 tightMarshal2 . . . . .	1381
6.252.3.7 tightUnmarshal . . . . .	1381
6.253activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	
Class Reference . . . . .	1382
6.253.1 Detailed Description . . . . .	1382
6.253.2 Constructor & Destructor Documentation . . . . .	1383
6.253.2.1 ConsumerControlMarshaller . . . . .	1383
6.253.2.2 ~ConsumerControlMarshaller . . . . .	1383
6.253.3 Member Function Documentation . . . . .	1383
6.253.3.1 createObject . . . . .	1383
6.253.3.2 getDataStructureType . . . . .	1383
6.253.3.3 looseMarshal . . . . .	1383

6.253.3.4 looseUnmarshal . . . . .	1384
6.253.3.5 tightMarshal1 . . . . .	1384
6.253.3.6 tightMarshal2 . . . . .	1385
6.253.3.7 tightUnmarshal . . . . .	1385
6.254activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	
Class Reference . . . . .	1386
6.254.1 Detailed Description . . . . .	1386
6.254.2 Constructor & Destructor Documentation . . . . .	1387
6.254.2.1 ConsumerControlMarshaller . . . . .	1387
6.254.2.2 ~ConsumerControlMarshaller . . . . .	1387
6.254.3 Member Function Documentation . . . . .	1387
6.254.3.1 createObject . . . . .	1387
6.254.3.2 getDataStructureType . . . . .	1387
6.254.3.3 looseMarshal . . . . .	1387
6.254.3.4 looseUnmarshal . . . . .	1388
6.254.3.5 tightMarshal1 . . . . .	1388
6.254.3.6 tightMarshal2 . . . . .	1389
6.254.3.7 tightUnmarshal . . . . .	1389
6.255activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	
Class Reference . . . . .	1390
6.255.1 Detailed Description . . . . .	1390
6.255.2 Constructor & Destructor Documentation . . . . .	1391
6.255.2.1 ConsumerControlMarshaller . . . . .	1391
6.255.2.2 ~ConsumerControlMarshaller . . . . .	1391
6.255.3 Member Function Documentation . . . . .	1391
6.255.3.1 createObject . . . . .	1391
6.255.3.2 getDataStructureType . . . . .	1391
6.255.3.3 looseMarshal . . . . .	1391
6.255.3.4 looseUnmarshal . . . . .	1392
6.255.3.5 tightMarshal1 . . . . .	1392
6.255.3.6 tightMarshal2 . . . . .	1393
6.255.3.7 tightUnmarshal . . . . .	1393
6.256activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller	
Class Reference . . . . .	1394
6.256.1 Detailed Description . . . . .	1394



6.256.2 Constructor & Destructor Documentation . . . . .	1395
6.256.2.1 ConsumerControlMarshaller . . . . .	1395
6.256.2.2 ~ConsumerControlMarshaller . . . . .	1395
6.256.3 Member Function Documentation . . . . .	1395
6.256.3.1 createObject . . . . .	1395
6.256.3.2 getDataStructureType . . . . .	1395
6.256.3.3 looseMarshal . . . . .	1395
6.256.3.4 looseUnmarshal . . . . .	1396
6.256.3.5 tightMarshal1 . . . . .	1396
6.256.3.6 tightMarshal2 . . . . .	1397
6.256.3.7 tightUnmarshal . . . . .	1397
6.257 activemq::commands::ConsumerId Class Reference . . . . .	1398
6.257.1 Member Typedef Documentation . . . . .	1399
6.257.1.1 COMPARATOR . . . . .	1399
6.257.2 Constructor & Destructor Documentation . . . . .	1399
6.257.2.1 ConsumerId . . . . .	1399
6.257.2.2 ConsumerId . . . . .	1399
6.257.2.3 ConsumerId . . . . .	1399
6.257.2.4 ~ConsumerId . . . . .	1399
6.257.3 Member Function Documentation . . . . .	1399
6.257.3.1 cloneDataStructure . . . . .	1399
6.257.3.2 compareTo . . . . .	1399
6.257.3.3 copyDataStructure . . . . .	1400
6.257.3.4 equals . . . . .	1400
6.257.3.5 equals . . . . .	1400
6.257.3.6 getConnectionId . . . . .	1400
6.257.3.7 getConnectionId . . . . .	1400
6.257.3.8 getDataStructureType . . . . .	1400
6.257.3.9 getParentId . . . . .	1400
6.257.3.10 getSessionId . . . . .	1401
6.257.3.11 getValue . . . . .	1401
6.257.3.12 operator< . . . . .	1401
6.257.3.13 operator= . . . . .	1401
6.257.3.14 operator== . . . . .	1401

6.257.3.15	setConnectionId . . . . .	1401
6.257.3.16	setSessionId . . . . .	1401
6.257.3.17	setValue . . . . .	1401
6.257.3.18	toString . . . . .	1401
6.257.4	Field Documentation . . . . .	1401
6.257.4.1	connectionId . . . . .	1401
6.257.4.2	ID_CONSUMERID . . . . .	1401
6.257.4.3	sessionId . . . . .	1402
6.257.4.4	value . . . . .	1402
6.258	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class	
	Reference . . . . .	1402
6.258.1	Detailed Description . . . . .	1403
6.258.2	Constructor & Destructor Documentation . . . . .	1403
6.258.2.1	ConsumerIdMarshaller . . . . .	1403
6.258.2.2	~ConsumerIdMarshaller . . . . .	1403
6.258.3	Member Function Documentation . . . . .	1403
6.258.3.1	createObject . . . . .	1403
6.258.3.2	getDataStructureType . . . . .	1403
6.258.3.3	looseMarshal . . . . .	1404
6.258.3.4	looseUnmarshal . . . . .	1404
6.258.3.5	tightMarshal1 . . . . .	1404
6.258.3.6	tightMarshal2 . . . . .	1405
6.258.3.7	tightUnmarshal . . . . .	1405
6.259	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class	
	Reference . . . . .	1406
6.259.1	Detailed Description . . . . .	1407
6.259.2	Constructor & Destructor Documentation . . . . .	1407
6.259.2.1	ConsumerIdMarshaller . . . . .	1407
6.259.2.2	~ConsumerIdMarshaller . . . . .	1407
6.259.3	Member Function Documentation . . . . .	1407
6.259.3.1	createObject . . . . .	1407
6.259.3.2	getDataStructureType . . . . .	1407
6.259.3.3	looseMarshal . . . . .	1408
6.259.3.4	looseUnmarshal . . . . .	1408

6.259.3.5 tightMarshal1 . . . . .	1408
6.259.3.6 tightMarshal2 . . . . .	1409
6.259.3.7 tightUnmarshal . . . . .	1409
6.260activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class	
Reference . . . . .	1410
6.260.1 Detailed Description . . . . .	1411
6.260.2 Constructor & Destructor Documentation . . . . .	1411
6.260.2.1 ConsumerIdMarshaller . . . . .	1411
6.260.2.2 ~ConsumerIdMarshaller . . . . .	1411
6.260.3 Member Function Documentation . . . . .	1411
6.260.3.1 createObject . . . . .	1411
6.260.3.2 getDataStructureType . . . . .	1411
6.260.3.3 looseMarshal . . . . .	1412
6.260.3.4 looseUnmarshal . . . . .	1412
6.260.3.5 tightMarshal1 . . . . .	1412
6.260.3.6 tightMarshal2 . . . . .	1413
6.260.3.7 tightUnmarshal . . . . .	1413
6.261activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class	
Reference . . . . .	1414
6.261.1 Detailed Description . . . . .	1415
6.261.2 Constructor & Destructor Documentation . . . . .	1415
6.261.2.1 ConsumerIdMarshaller . . . . .	1415
6.261.2.2 ~ConsumerIdMarshaller . . . . .	1415
6.261.3 Member Function Documentation . . . . .	1415
6.261.3.1 createObject . . . . .	1415
6.261.3.2 getDataStructureType . . . . .	1415
6.261.3.3 looseMarshal . . . . .	1416
6.261.3.4 looseUnmarshal . . . . .	1416
6.261.3.5 tightMarshal1 . . . . .	1416
6.261.3.6 tightMarshal2 . . . . .	1417
6.261.3.7 tightUnmarshal . . . . .	1417
6.262activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class	
Reference . . . . .	1418
6.262.1 Detailed Description . . . . .	1419
6.262.2 Constructor & Destructor Documentation . . . . .	1419

6.262.2.1 ConsumerIdMarshaller . . . . .	1419
6.262.2.2 ~ConsumerIdMarshaller . . . . .	1419
6.262.3 Member Function Documentation . . . . .	1419
6.262.3.1 createObject . . . . .	1419
6.262.3.2 getDataStructureType . . . . .	1419
6.262.3.3 looseMarshal . . . . .	1420
6.262.3.4 looseUnmarshal . . . . .	1420
6.262.3.5 tightMarshal1 . . . . .	1420
6.262.3.6 tightMarshal2 . . . . .	1421
6.262.3.7 tightUnmarshal . . . . .	1421
6.263activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller Class Reference . . . . .	1422
6.263.1 Detailed Description . . . . .	1423
6.263.2 Constructor & Destructor Documentation . . . . .	1423
6.263.2.1 ConsumerIdMarshaller . . . . .	1423
6.263.2.2 ~ConsumerIdMarshaller . . . . .	1423
6.263.3 Member Function Documentation . . . . .	1423
6.263.3.1 createObject . . . . .	1423
6.263.3.2 getDataStructureType . . . . .	1423
6.263.3.3 looseMarshal . . . . .	1424
6.263.3.4 looseUnmarshal . . . . .	1424
6.263.3.5 tightMarshal1 . . . . .	1424
6.263.3.6 tightMarshal2 . . . . .	1425
6.263.3.7 tightUnmarshal . . . . .	1425
6.264activemq::commands::ConsumerInfo Class Reference . . . . .	1426
6.264.1 Constructor & Destructor Documentation . . . . .	1428
6.264.1.1 ConsumerInfo . . . . .	1428
6.264.1.2 ~ConsumerInfo . . . . .	1428
6.264.2 Member Function Documentation . . . . .	1428
6.264.2.1 cloneDataStructure . . . . .	1428
6.264.2.2 copyDataStructure . . . . .	1429
6.264.2.3 createRemoveCommand . . . . .	1429
6.264.2.4 equals . . . . .	1429
6.264.2.5 getAdditionalPredicate . . . . .	1429

6.264.2.6	getAdditionalPredicate . . . . .	1429
6.264.2.7	getBrokerPath . . . . .	1429
6.264.2.8	getBrokerPath . . . . .	1429
6.264.2.9	getConsumerId . . . . .	1429
6.264.2.10	getConsumerId . . . . .	1430
6.264.2.11	getDataStructureType . . . . .	1430
6.264.2.12	getDestination . . . . .	1430
6.264.2.13	getDestination . . . . .	1430
6.264.2.14	getMaximumPendingMessageLimit . . . . .	1430
6.264.2.15	getNetworkConsumerPath . . . . .	1430
6.264.2.16	getNetworkConsumerPath . . . . .	1430
6.264.2.17	getPrefetchSize . . . . .	1430
6.264.2.18	getPriority . . . . .	1430
6.264.2.19	getSelector . . . . .	1430
6.264.2.20	getSelector . . . . .	1430
6.264.2.21	getSubscriptionName . . . . .	1430
6.264.2.22	getSubscriptionName . . . . .	1431
6.264.2.23	isBrowser . . . . .	1431
6.264.2.24	isConsumerInfo . . . . .	1431
6.264.2.25	isDispatchAsync . . . . .	1431
6.264.2.26	isExclusive . . . . .	1431
6.264.2.27	isNetworkSubscription . . . . .	1431
6.264.2.28	isNoLocal . . . . .	1431
6.264.2.29	isNoRangeAcks . . . . .	1431
6.264.2.30	isOptimizedAcknowledge . . . . .	1431
6.264.2.31	isRetroactive . . . . .	1431
6.264.2.32	setAdditionalPredicate . . . . .	1431
6.264.2.33	setBrokerPath . . . . .	1431
6.264.2.34	setBrowser . . . . .	1431
6.264.2.35	setConsumerId . . . . .	1431
6.264.2.36	setDestination . . . . .	1432
6.264.2.37	setDispatchAsync . . . . .	1432
6.264.2.38	setExclusive . . . . .	1432
6.264.2.39	setMaximumPendingMessageLimit . . . . .	1432

6.264.2.40	setNetworkConsumerPath . . . . .	1432
6.264.2.41	setNetworkSubscription . . . . .	1432
6.264.2.42	setNoLocal . . . . .	1432
6.264.2.43	setNoRangeAcks . . . . .	1432
6.264.2.44	setOptimizedAcknowledge . . . . .	1432
6.264.2.45	setPrefetchSize . . . . .	1432
6.264.2.46	setPriority . . . . .	1432
6.264.2.47	setRetroactive . . . . .	1432
6.264.2.48	setSelector . . . . .	1432
6.264.2.49	setSubscriptionName . . . . .	1432
6.264.2.50	toString . . . . .	1432
6.264.2.51	visit . . . . .	1433
6.264.3	Field Documentation . . . . .	1433
6.264.3.1	additionalPredicate . . . . .	1433
6.264.3.2	brokerPath . . . . .	1433
6.264.3.3	browser . . . . .	1433
6.264.3.4	consumerId . . . . .	1433
6.264.3.5	destination . . . . .	1433
6.264.3.6	dispatchAsync . . . . .	1433
6.264.3.7	exclusive . . . . .	1433
6.264.3.8	ID_CONSUMERINFO . . . . .	1433
6.264.3.9	maximumPendingMessageLimit . . . . .	1433
6.264.3.10	networkConsumerPath . . . . .	1434
6.264.3.11	networkSubscription . . . . .	1434
6.264.3.12	noLocal . . . . .	1434
6.264.3.13	noRangeAcks . . . . .	1434
6.264.3.14	optimizedAcknowledge . . . . .	1434
6.264.3.15	prefetchSize . . . . .	1434
6.264.3.16	priority . . . . .	1434
6.264.3.17	retroactive . . . . .	1434
6.264.3.18	selector . . . . .	1434
6.264.3.19	subscriptionName . . . . .	1434
6.265	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller Class Reference . . . . .	1434

6.265.1 Detailed Description . . . . .	1435
6.265.2 Constructor & Destructor Documentation . . . . .	1435
6.265.2.1 ConsumerInfoMarshaller . . . . .	1435
6.265.2.2 ~ConsumerInfoMarshaller . . . . .	1435
6.265.3 Member Function Documentation . . . . .	1435
6.265.3.1 createObject . . . . .	1436
6.265.3.2 getDataStructureType . . . . .	1436
6.265.3.3 looseMarshal . . . . .	1436
6.265.3.4 looseUnmarshal . . . . .	1437
6.265.3.5 tightMarshal1 . . . . .	1437
6.265.3.6 tightMarshal2 . . . . .	1438
6.265.3.7 tightUnmarshal . . . . .	1438
6.266activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference . . . . .	1439
6.266.1 Detailed Description . . . . .	1439
6.266.2 Constructor & Destructor Documentation . . . . .	1440
6.266.2.1 ConsumerInfoMarshaller . . . . .	1440
6.266.2.2 ~ConsumerInfoMarshaller . . . . .	1440
6.266.3 Member Function Documentation . . . . .	1440
6.266.3.1 createObject . . . . .	1440
6.266.3.2 getDataStructureType . . . . .	1440
6.266.3.3 looseMarshal . . . . .	1440
6.266.3.4 looseUnmarshal . . . . .	1441
6.266.3.5 tightMarshal1 . . . . .	1441
6.266.3.6 tightMarshal2 . . . . .	1442
6.266.3.7 tightUnmarshal . . . . .	1442
6.267activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference . . . . .	1443
6.267.1 Detailed Description . . . . .	1443
6.267.2 Constructor & Destructor Documentation . . . . .	1444
6.267.2.1 ConsumerInfoMarshaller . . . . .	1444
6.267.2.2 ~ConsumerInfoMarshaller . . . . .	1444
6.267.3 Member Function Documentation . . . . .	1444
6.267.3.1 createObject . . . . .	1444

6.267.3.2	getDataStructureType . . . . .	1444
6.267.3.3	looseMarshal . . . . .	1444
6.267.3.4	looseUnmarshal . . . . .	1445
6.267.3.5	tightMarshal1 . . . . .	1445
6.267.3.6	tightMarshal2 . . . . .	1446
6.267.3.7	tightUnmarshal . . . . .	1446
6.268	activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class	
	Reference . . . . .	1447
6.268.1	Detailed Description . . . . .	1447
6.268.2	Constructor & Destructor Documentation . . . . .	1448
6.268.2.1	ConsumerInfoMarshaller . . . . .	1448
6.268.2.2	~ConsumerInfoMarshaller . . . . .	1448
6.268.3	Member Function Documentation . . . . .	1448
6.268.3.1	createObject . . . . .	1448
6.268.3.2	getDataStructureType . . . . .	1448
6.268.3.3	looseMarshal . . . . .	1448
6.268.3.4	looseUnmarshal . . . . .	1449
6.268.3.5	tightMarshal1 . . . . .	1449
6.268.3.6	tightMarshal2 . . . . .	1450
6.268.3.7	tightUnmarshal . . . . .	1450
6.269	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class	
	Reference . . . . .	1451
6.269.1	Detailed Description . . . . .	1451
6.269.2	Constructor & Destructor Documentation . . . . .	1452
6.269.2.1	ConsumerInfoMarshaller . . . . .	1452
6.269.2.2	~ConsumerInfoMarshaller . . . . .	1452
6.269.3	Member Function Documentation . . . . .	1452
6.269.3.1	createObject . . . . .	1452
6.269.3.2	getDataStructureType . . . . .	1452
6.269.3.3	looseMarshal . . . . .	1452
6.269.3.4	looseUnmarshal . . . . .	1453
6.269.3.5	tightMarshal1 . . . . .	1453
6.269.3.6	tightMarshal2 . . . . .	1454
6.269.3.7	tightUnmarshal . . . . .	1454



6.270activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller Class Reference . . . . .	1455
6.270.1 Detailed Description . . . . .	1455
6.270.2 Constructor & Destructor Documentation . . . . .	1456
6.270.2.1 ConsumerInfoMarshaller . . . . .	1456
6.270.2.2 ~ConsumerInfoMarshaller . . . . .	1456
6.270.3 Member Function Documentation . . . . .	1456
6.270.3.1 createObject . . . . .	1456
6.270.3.2 getDataStructureType . . . . .	1456
6.270.3.3 looseMarshal . . . . .	1456
6.270.3.4 looseUnmarshal . . . . .	1457
6.270.3.5 tightMarshal1 . . . . .	1457
6.270.3.6 tightMarshal2 . . . . .	1458
6.270.3.7 tightUnmarshal . . . . .	1458
6.271activemq::state::ConsumerState Class Reference . . . . .	1459
6.271.1 Constructor & Destructor Documentation . . . . .	1459
6.271.1.1 ConsumerState . . . . .	1459
6.271.1.2 ~ConsumerState . . . . .	1459
6.271.2 Member Function Documentation . . . . .	1459
6.271.2.1 getInfo . . . . .	1459
6.271.2.2 toString . . . . .	1459
6.272activemq::commands::ControlCommand Class Reference . . . . .	1459
6.272.1 Constructor & Destructor Documentation . . . . .	1460
6.272.1.1 ControlCommand . . . . .	1460
6.272.1.2 ~ControlCommand . . . . .	1460
6.272.2 Member Function Documentation . . . . .	1460
6.272.2.1 cloneDataStructure . . . . .	1460
6.272.2.2 copyDataStructure . . . . .	1461
6.272.2.3 equals . . . . .	1461
6.272.2.4 getCommand . . . . .	1461
6.272.2.5 getCommand . . . . .	1461
6.272.2.6 getDataStructureType . . . . .	1461
6.272.2.7 setCommand . . . . .	1461
6.272.2.8 toString . . . . .	1462

6.272.2.9 visit . . . . .	1462
6.272.3 Field Documentation . . . . .	1462
6.272.3.1 command . . . . .	1462
6.272.3.2 ID_CONTROLCOMMAND . . . . .	1462
6.273activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	
Class Reference . . . . .	1462
6.273.1 Detailed Description . . . . .	1463
6.273.2 Constructor & Destructor Documentation . . . . .	1463
6.273.2.1 ControlCommandMarshaller . . . . .	1463
6.273.2.2 ~ControlCommandMarshaller . . . . .	1464
6.273.3 Member Function Documentation . . . . .	1464
6.273.3.1 createObject . . . . .	1464
6.273.3.2 getDataStructureType . . . . .	1464
6.273.3.3 looseMarshal . . . . .	1464
6.273.3.4 looseUnmarshal . . . . .	1465
6.273.3.5 tightMarshal1 . . . . .	1465
6.273.3.6 tightMarshal2 . . . . .	1466
6.273.3.7 tightUnmarshal . . . . .	1466
6.274activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller	
Class Reference . . . . .	1467
6.274.1 Detailed Description . . . . .	1467
6.274.2 Constructor & Destructor Documentation . . . . .	1468
6.274.2.1 ControlCommandMarshaller . . . . .	1468
6.274.2.2 ~ControlCommandMarshaller . . . . .	1468
6.274.3 Member Function Documentation . . . . .	1468
6.274.3.1 createObject . . . . .	1468
6.274.3.2 getDataStructureType . . . . .	1468
6.274.3.3 looseMarshal . . . . .	1468
6.274.3.4 looseUnmarshal . . . . .	1469
6.274.3.5 tightMarshal1 . . . . .	1469
6.274.3.6 tightMarshal2 . . . . .	1470
6.274.3.7 tightUnmarshal . . . . .	1470
6.275activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller	
Class Reference . . . . .	1471
6.275.1 Detailed Description . . . . .	1471

6.275.2 Constructor & Destructor Documentation . . . . .	1472
6.275.2.1 ControlCommandMarshaller . . . . .	1472
6.275.2.2 ~ControlCommandMarshaller . . . . .	1472
6.275.3 Member Function Documentation . . . . .	1472
6.275.3.1 createObject . . . . .	1472
6.275.3.2 getDataStructureType . . . . .	1472
6.275.3.3 looseMarshal . . . . .	1472
6.275.3.4 looseUnmarshal . . . . .	1473
6.275.3.5 tightMarshal1 . . . . .	1473
6.275.3.6 tightMarshal2 . . . . .	1474
6.275.3.7 tightUnmarshal . . . . .	1474
6.276activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller	
Class Reference . . . . .	1475
6.276.1 Detailed Description . . . . .	1475
6.276.2 Constructor & Destructor Documentation . . . . .	1476
6.276.2.1 ControlCommandMarshaller . . . . .	1476
6.276.2.2 ~ControlCommandMarshaller . . . . .	1476
6.276.3 Member Function Documentation . . . . .	1476
6.276.3.1 createObject . . . . .	1476
6.276.3.2 getDataStructureType . . . . .	1476
6.276.3.3 looseMarshal . . . . .	1476
6.276.3.4 looseUnmarshal . . . . .	1477
6.276.3.5 tightMarshal1 . . . . .	1477
6.276.3.6 tightMarshal2 . . . . .	1478
6.276.3.7 tightUnmarshal . . . . .	1478
6.277activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller	
Class Reference . . . . .	1479
6.277.1 Detailed Description . . . . .	1479
6.277.2 Constructor & Destructor Documentation . . . . .	1480
6.277.2.1 ControlCommandMarshaller . . . . .	1480
6.277.2.2 ~ControlCommandMarshaller . . . . .	1480
6.277.3 Member Function Documentation . . . . .	1480
6.277.3.1 createObject . . . . .	1480
6.277.3.2 getDataStructureType . . . . .	1480

6.277.3.3 looseMarshal . . . . .	1480
6.277.3.4 looseUnmarshal . . . . .	1481
6.277.3.5 tightMarshal1 . . . . .	1481
6.277.3.6 tightMarshal2 . . . . .	1482
6.277.3.7 tightUnmarshal . . . . .	1482
6.278activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller Class Reference . . . . .	1483
6.278.1 Detailed Description . . . . .	1483
6.278.2 Constructor & Destructor Documentation . . . . .	1484
6.278.2.1 ControlCommandMarshaller . . . . .	1484
6.278.2.2 ~ControlCommandMarshaller . . . . .	1484
6.278.3 Member Function Documentation . . . . .	1484
6.278.3.1 createObject . . . . .	1484
6.278.3.2 getDataStructureType . . . . .	1484
6.278.3.3 looseMarshal . . . . .	1484
6.278.3.4 looseUnmarshal . . . . .	1485
6.278.3.5 tightMarshal1 . . . . .	1485
6.278.3.6 tightMarshal2 . . . . .	1486
6.278.3.7 tightUnmarshal . . . . .	1486
6.279decaf::util::concurrent::CountDownLatch Class Reference . . . . .	1487
6.279.1 Constructor & Destructor Documentation . . . . .	1487
6.279.1.1 CountDownLatch . . . . .	1487
6.279.1.2 ~CountDownLatch . . . . .	1487
6.279.2 Member Function Documentation . . . . .	1487
6.279.2.1 await . . . . .	1488
6.279.2.2 await . . . . .	1488
6.279.2.3 await . . . . .	1489
6.279.2.4 countDown . . . . .	1489
6.279.2.5 getCount . . . . .	1489
6.280decaf::util::zip::CRC32 Class Reference . . . . .	1490
6.280.1 Detailed Description . . . . .	1490
6.280.2 Constructor & Destructor Documentation . . . . .	1491
6.280.2.1 CRC32 . . . . .	1491
6.280.2.2 ~CRC32 . . . . .	1491

6.280.3 Member Function Documentation . . . . .	1491
6.280.3.1 getValue . . . . .	1491
6.280.3.2 reset . . . . .	1491
6.280.3.3 update . . . . .	1491
6.280.3.4 update . . . . .	1491
6.280.3.5 update . . . . .	1492
6.280.3.6 update . . . . .	1492
6.281 ct_data_s Struct Reference . . . . .	1492
6.281.1 Field Documentation . . . . .	1493
6.281.1.1 code . . . . .	1493
6.281.1.2 dad . . . . .	1493
6.281.1.3 dl . . . . .	1493
6.281.1.4 fc . . . . .	1493
6.281.1.5 freq . . . . .	1493
6.281.1.6 len . . . . .	1493
6.282 activemq::commands::DataArrayResponse Class Reference . . . . .	1493
6.282.1 Constructor & Destructor Documentation . . . . .	1494
6.282.1.1 DataArrayResponse . . . . .	1494
6.282.1.2 ~DataArrayResponse . . . . .	1494
6.282.2 Member Function Documentation . . . . .	1494
6.282.2.1 cloneDataStructure . . . . .	1494
6.282.2.2 copyDataStructure . . . . .	1494
6.282.2.3 equals . . . . .	1495
6.282.2.4 getData . . . . .	1495
6.282.2.5 getData . . . . .	1495
6.282.2.6 getDataStructureType . . . . .	1495
6.282.2.7 setData . . . . .	1495
6.282.2.8 toString . . . . .	1495
6.282.3 Field Documentation . . . . .	1496
6.282.3.1 data . . . . .	1496
6.282.3.2 ID_DATAARRAYRESPONSE . . . . .	1496
6.283 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class Reference . . . . .	1496
6.283.1 Detailed Description . . . . .	1497

6.283.2 Constructor & Destructor Documentation . . . . .	1497
6.283.2.1 DataArrayResponseMarshaller . . . . .	1497
6.283.2.2 ~DataArrayResponseMarshaller . . . . .	1497
6.283.3 Member Function Documentation . . . . .	1497
6.283.3.1 createObject . . . . .	1497
6.283.3.2 getDataStructureType . . . . .	1497
6.283.3.3 looseMarshal . . . . .	1498
6.283.3.4 looseUnmarshal . . . . .	1498
6.283.3.5 tightMarshal1 . . . . .	1498
6.283.3.6 tightMarshal2 . . . . .	1499
6.283.3.7 tightUnmarshal . . . . .	1499
6.284activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller	
Class Reference . . . . .	1500
6.284.1 Detailed Description . . . . .	1501
6.284.2 Constructor & Destructor Documentation . . . . .	1501
6.284.2.1 DataArrayResponseMarshaller . . . . .	1501
6.284.2.2 ~DataArrayResponseMarshaller . . . . .	1501
6.284.3 Member Function Documentation . . . . .	1501
6.284.3.1 createObject . . . . .	1501
6.284.3.2 getDataStructureType . . . . .	1501
6.284.3.3 looseMarshal . . . . .	1502
6.284.3.4 looseUnmarshal . . . . .	1502
6.284.3.5 tightMarshal1 . . . . .	1503
6.284.3.6 tightMarshal2 . . . . .	1503
6.284.3.7 tightUnmarshal . . . . .	1504
6.285activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	
Class Reference . . . . .	1504
6.285.1 Detailed Description . . . . .	1505
6.285.2 Constructor & Destructor Documentation . . . . .	1505
6.285.2.1 DataArrayResponseMarshaller . . . . .	1505
6.285.2.2 ~DataArrayResponseMarshaller . . . . .	1505
6.285.3 Member Function Documentation . . . . .	1505
6.285.3.1 createObject . . . . .	1505
6.285.3.2 getDataStructureType . . . . .	1506

6.285.3.3 looseMarshal . . . . .	1506
6.285.3.4 looseUnmarshal . . . . .	1506
6.285.3.5 tightMarshal1 . . . . .	1507
6.285.3.6 tightMarshal2 . . . . .	1507
6.285.3.7 tightUnmarshal . . . . .	1508
6.286activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class Reference . . . . .	1508
6.286.1 Detailed Description . . . . .	1509
6.286.2 Constructor & Destructor Documentation . . . . .	1509
6.286.2.1 DataArrayResponseMarshaller . . . . .	1509
6.286.2.2 ~DataArrayResponseMarshaller . . . . .	1509
6.286.3 Member Function Documentation . . . . .	1509
6.286.3.1 createObject . . . . .	1509
6.286.3.2 getDataStructureType . . . . .	1510
6.286.3.3 looseMarshal . . . . .	1510
6.286.3.4 looseUnmarshal . . . . .	1510
6.286.3.5 tightMarshal1 . . . . .	1511
6.286.3.6 tightMarshal2 . . . . .	1511
6.286.3.7 tightUnmarshal . . . . .	1512
6.287activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class Reference . . . . .	1512
6.287.1 Detailed Description . . . . .	1513
6.287.2 Constructor & Destructor Documentation . . . . .	1513
6.287.2.1 DataArrayResponseMarshaller . . . . .	1513
6.287.2.2 ~DataArrayResponseMarshaller . . . . .	1513
6.287.3 Member Function Documentation . . . . .	1513
6.287.3.1 createObject . . . . .	1513
6.287.3.2 getDataStructureType . . . . .	1514
6.287.3.3 looseMarshal . . . . .	1514
6.287.3.4 looseUnmarshal . . . . .	1514
6.287.3.5 tightMarshal1 . . . . .	1515
6.287.3.6 tightMarshal2 . . . . .	1515
6.287.3.7 tightUnmarshal . . . . .	1516
6.288activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference . . . . .	1516

6.288.1 Detailed Description . . . . .	1517
6.288.2 Constructor & Destructor Documentation . . . . .	1517
6.288.2.1 DataArrayResponseMarshaller . . . . .	1517
6.288.2.2 ~DataArrayResponseMarshaller . . . . .	1517
6.288.3 Member Function Documentation . . . . .	1517
6.288.3.1 createObject . . . . .	1517
6.288.3.2 getDataStructureType . . . . .	1518
6.288.3.3 looseMarshal . . . . .	1518
6.288.3.4 looseUnmarshal . . . . .	1518
6.288.3.5 tightMarshal1 . . . . .	1519
6.288.3.6 tightMarshal2 . . . . .	1519
6.288.3.7 tightUnmarshal . . . . .	1520
6.289decaf::util::zip::DataFormatException Class Reference . . . . .	1520
6.289.1 Constructor & Destructor Documentation . . . . .	1521
6.289.1.1 DataFormatException . . . . .	1521
6.289.1.2 DataFormatException . . . . .	1521
6.289.1.3 DataFormatException . . . . .	1521
6.289.1.4 DataFormatException . . . . .	1521
6.289.1.5 DataFormatException . . . . .	1522
6.289.1.6 DataFormatException . . . . .	1522
6.289.1.7 ~DataFormatException . . . . .	1522
6.289.2 Member Function Documentation . . . . .	1522
6.289.2.1 clone . . . . .	1522
6.290decaf::io::DataInput Class Reference . . . . .	1523
6.290.1 Detailed Description . . . . .	1524
6.290.2 Constructor & Destructor Documentation . . . . .	1524
6.290.2.1 ~DataInput . . . . .	1524
6.290.3 Member Function Documentation . . . . .	1524
6.290.3.1 readBoolean . . . . .	1525
6.290.3.2 readByte . . . . .	1525
6.290.3.3 readChar . . . . .	1525
6.290.3.4 readDouble . . . . .	1526
6.290.3.5 readFloat . . . . .	1526
6.290.3.6 readFully . . . . .	1526



6.290.3.7 readFully . . . . .	1527
6.290.3.8 readInt . . . . .	1528
6.290.3.9 readLine . . . . .	1528
6.290.3.10 readLong . . . . .	1529
6.290.3.11 readShort . . . . .	1529
6.290.3.12 readString . . . . .	1530
6.290.3.13 readUnsignedByte . . . . .	1530
6.290.3.14 readUnsignedShort . . . . .	1530
6.290.3.15 readUTF . . . . .	1531
6.290.3.16 skipBytes . . . . .	1531
6.291 decaf::io::DataInputStream Class Reference . . . . .	1532
6.291.1 Detailed Description . . . . .	1533
6.291.2 Constructor & Destructor Documentation . . . . .	1534
6.291.2.1 DataInputStream . . . . .	1534
6.291.2.2 ~DataInputStream . . . . .	1534
6.291.3 Member Function Documentation . . . . .	1534
6.291.3.1 readBoolean . . . . .	1534
6.291.3.2 readByte . . . . .	1534
6.291.3.3 readChar . . . . .	1535
6.291.3.4 readDouble . . . . .	1535
6.291.3.5 readFloat . . . . .	1535
6.291.3.6 readFully . . . . .	1536
6.291.3.7 readFully . . . . .	1536
6.291.3.8 readInt . . . . .	1537
6.291.3.9 readLine . . . . .	1537
6.291.3.10 readLong . . . . .	1538
6.291.3.11 readShort . . . . .	1538
6.291.3.12 readString . . . . .	1539
6.291.3.13 readUnsignedByte . . . . .	1539
6.291.3.14 readUnsignedShort . . . . .	1540
6.291.3.15 readUTF . . . . .	1540
6.291.3.16 skipBytes . . . . .	1540
6.292 decaf::io::DataOutput Class Reference . . . . .	1541
6.292.1 Detailed Description . . . . .	1542

6.292.2 Constructor & Destructor Documentation . . . . .	1542
6.292.2.1 ~DataOutput . . . . .	1542
6.292.3 Member Function Documentation . . . . .	1542
6.292.3.1 writeBoolean . . . . .	1542
6.292.3.2 writeByte . . . . .	1543
6.292.3.3 writeBytes . . . . .	1543
6.292.3.4 writeChar . . . . .	1543
6.292.3.5 writeChars . . . . .	1544
6.292.3.6 writeDouble . . . . .	1544
6.292.3.7 writeFloat . . . . .	1544
6.292.3.8 writeInt . . . . .	1545
6.292.3.9 writeLong . . . . .	1545
6.292.3.10writeShort . . . . .	1545
6.292.3.11writeUnsignedShort . . . . .	1546
6.292.3.12writeUTF . . . . .	1546
6.293decaf::io::DataOutputStream Class Reference . . . . .	1546
6.293.1 Detailed Description . . . . .	1548
6.293.2 Constructor & Destructor Documentation . . . . .	1548
6.293.2.1 DataOutputStream . . . . .	1548
6.293.2.2 ~DataOutputStream . . . . .	1548
6.293.3 Member Function Documentation . . . . .	1548
6.293.3.1 doWriteArrayBounded . . . . .	1548
6.293.3.2 doWriteByte . . . . .	1549
6.293.3.3 size . . . . .	1549
6.293.3.4 writeBoolean . . . . .	1549
6.293.3.5 writeByte . . . . .	1549
6.293.3.6 writeBytes . . . . .	1549
6.293.3.7 writeChar . . . . .	1549
6.293.3.8 writeChars . . . . .	1549
6.293.3.9 writeDouble . . . . .	1549
6.293.3.10writeFloat . . . . .	1549
6.293.3.11writeInt . . . . .	1549
6.293.3.12writeLong . . . . .	1549
6.293.3.13writeShort . . . . .	1549

6.293.3.14	writeUnsignedShort . . . . .	1550
6.293.3.15	writeUTF . . . . .	1550
6.293.4	Field Documentation . . . . .	1550
6.293.4.1	buffer . . . . .	1550
6.293.4.2	written . . . . .	1550
6.294	activemq::commands::DataResponse Class Reference . . . . .	1550
6.294.1	Constructor & Destructor Documentation . . . . .	1551
6.294.1.1	DataResponse . . . . .	1551
6.294.1.2	~DataResponse . . . . .	1551
6.294.2	Member Function Documentation . . . . .	1551
6.294.2.1	cloneDataStructure . . . . .	1551
6.294.2.2	copyDataStructure . . . . .	1551
6.294.2.3	equals . . . . .	1551
6.294.2.4	getData . . . . .	1552
6.294.2.5	getData . . . . .	1552
6.294.2.6	getDataStructureType . . . . .	1552
6.294.2.7	setData . . . . .	1552
6.294.2.8	toString . . . . .	1552
6.294.3	Field Documentation . . . . .	1552
6.294.3.1	data . . . . .	1552
6.294.3.2	ID_DATARESPONSE . . . . .	1552
6.295	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Reference . . . . .	1553
6.295.1	Detailed Description . . . . .	1554
6.295.2	Constructor & Destructor Documentation . . . . .	1554
6.295.2.1	DataResponseMarshaller . . . . .	1554
6.295.2.2	~DataResponseMarshaller . . . . .	1554
6.295.3	Member Function Documentation . . . . .	1554
6.295.3.1	createObject . . . . .	1554
6.295.3.2	getDataStructureType . . . . .	1554
6.295.3.3	looseMarshal . . . . .	1554
6.295.3.4	looseUnmarshal . . . . .	1555
6.295.3.5	tightMarshal1 . . . . .	1555
6.295.3.6	tightMarshal2 . . . . .	1556

6.295.3.7 tightUnmarshal . . . . .	1556
6.296activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller Class Reference . . . . .	1557
6.296.1 Detailed Description . . . . .	1558
6.296.2 Constructor & Destructor Documentation . . . . .	1558
6.296.2.1 DataResponseMarshaller . . . . .	1558
6.296.2.2 ~DataResponseMarshaller . . . . .	1558
6.296.3 Member Function Documentation . . . . .	1558
6.296.3.1 createObject . . . . .	1558
6.296.3.2 getDataStructureType . . . . .	1558
6.296.3.3 looseMarshal . . . . .	1559
6.296.3.4 looseUnmarshal . . . . .	1559
6.296.3.5 tightMarshal1 . . . . .	1559
6.296.3.6 tightMarshal2 . . . . .	1560
6.296.3.7 tightUnmarshal . . . . .	1560
6.297activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference . . . . .	1561
6.297.1 Detailed Description . . . . .	1562
6.297.2 Constructor & Destructor Documentation . . . . .	1562
6.297.2.1 DataResponseMarshaller . . . . .	1562
6.297.2.2 ~DataResponseMarshaller . . . . .	1562
6.297.3 Member Function Documentation . . . . .	1562
6.297.3.1 createObject . . . . .	1562
6.297.3.2 getDataStructureType . . . . .	1562
6.297.3.3 looseMarshal . . . . .	1563
6.297.3.4 looseUnmarshal . . . . .	1563
6.297.3.5 tightMarshal1 . . . . .	1564
6.297.3.6 tightMarshal2 . . . . .	1564
6.297.3.7 tightUnmarshal . . . . .	1565
6.298activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller Class Reference . . . . .	1565
6.298.1 Detailed Description . . . . .	1566
6.298.2 Constructor & Destructor Documentation . . . . .	1566
6.298.2.1 DataResponseMarshaller . . . . .	1566
6.298.2.2 ~DataResponseMarshaller . . . . .	1566

6.298.3 Member Function Documentation . . . . .	1566
6.298.3.1 createObject . . . . .	1566
6.298.3.2 getDataStructureType . . . . .	1567
6.298.3.3 looseMarshal . . . . .	1567
6.298.3.4 looseUnmarshal . . . . .	1567
6.298.3.5 tightMarshal1 . . . . .	1568
6.298.3.6 tightMarshal2 . . . . .	1568
6.298.3.7 tightUnmarshal . . . . .	1569
6.299activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference . . . . .	1569
6.299.1 Detailed Description . . . . .	1570
6.299.2 Constructor & Destructor Documentation . . . . .	1570
6.299.2.1 DataResponseMarshaller . . . . .	1570
6.299.2.2 ~DataResponseMarshaller . . . . .	1570
6.299.3 Member Function Documentation . . . . .	1570
6.299.3.1 createObject . . . . .	1570
6.299.3.2 getDataStructureType . . . . .	1571
6.299.3.3 looseMarshal . . . . .	1571
6.299.3.4 looseUnmarshal . . . . .	1571
6.299.3.5 tightMarshal1 . . . . .	1572
6.299.3.6 tightMarshal2 . . . . .	1572
6.299.3.7 tightUnmarshal . . . . .	1573
6.300activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference . . . . .	1573
6.300.1 Detailed Description . . . . .	1574
6.300.2 Constructor & Destructor Documentation . . . . .	1574
6.300.2.1 DataResponseMarshaller . . . . .	1574
6.300.2.2 ~DataResponseMarshaller . . . . .	1574
6.300.3 Member Function Documentation . . . . .	1574
6.300.3.1 createObject . . . . .	1574
6.300.3.2 getDataStructureType . . . . .	1575
6.300.3.3 looseMarshal . . . . .	1575
6.300.3.4 looseUnmarshal . . . . .	1575
6.300.3.5 tightMarshal1 . . . . .	1576

6.300.3.6 tightMarshal2 . . . . .	1576
6.300.3.7 tightUnmarshal . . . . .	1577
6.301activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference . . . . .	1577
6.301.1 Detailed Description . . . . .	1578
6.301.2 Constructor & Destructor Documentation . . . . .	1578
6.301.2.1 ~DataStreamMarshaller . . . . .	1578
6.301.3 Member Function Documentation . . . . .	1578
6.301.3.1 createObject . . . . .	1578
6.301.3.2 getDataStructureType . . . . .	1585
6.301.3.3 looseMarshal . . . . .	1591
6.301.3.4 looseUnmarshal . . . . .	1599
6.301.3.5 tightMarshal1 . . . . .	1606
6.301.3.6 tightMarshal2 . . . . .	1613
6.301.3.7 tightUnmarshal . . . . .	1620
6.302activemq::commands::DataStructure Class Reference . . . . .	1628
6.302.1 Constructor & Destructor Documentation . . . . .	1628
6.302.1.1 ~DataStructure . . . . .	1628
6.302.2 Member Function Documentation . . . . .	1628
6.302.2.1 cloneDataStructure . . . . .	1628
6.302.2.2 copyDataStructure . . . . .	1629
6.302.2.3 equals . . . . .	1630
6.302.2.4 getDataStructureType . . . . .	1631
6.302.2.5 toString . . . . .	1632
6.303decaf::util::Date Class Reference . . . . .	1633
6.303.1 Detailed Description . . . . .	1634
6.303.2 Constructor & Destructor Documentation . . . . .	1634
6.303.2.1 Date . . . . .	1634
6.303.2.2 Date . . . . .	1634
6.303.2.3 Date . . . . .	1634
6.303.2.4 ~Date . . . . .	1635
6.303.3 Member Function Documentation . . . . .	1635
6.303.3.1 after . . . . .	1635
6.303.3.2 before . . . . .	1635

6.303.3.3	compareTo	1635
6.303.3.4	equals	1635
6.303.3.5	getTime	1636
6.303.3.6	operator<	1636
6.303.3.7	operator=	1636
6.303.3.8	operator==	1636
6.303.3.9	setTime	1637
6.303.3.10	toString	1637
6.304	decaf::internal::DecafRuntime Class Reference	1637
6.304.1	Detailed Description	1638
6.304.2	Constructor & Destructor Documentation	1638
6.304.2.1	DecafRuntime	1638
6.304.2.2	~DecafRuntime	1638
6.304.3	Member Function Documentation	1638
6.304.3.1	getGlobalPool	1638
6.305	activemq::threads::DedicatedTaskRunner Class Reference	1638
6.305.1	Constructor & Destructor Documentation	1639
6.305.1.1	DedicatedTaskRunner	1639
6.305.1.2	~DedicatedTaskRunner	1639
6.305.2	Member Function Documentation	1639
6.305.2.1	run	1639
6.305.2.2	shutdown	1639
6.305.2.3	shutdown	1639
6.305.2.4	wakeup	1640
6.306	activemq::core::policies::DefaultPrefetchPolicy Class Reference	1640
6.306.1	Constructor & Destructor Documentation	1641
6.306.1.1	DefaultPrefetchPolicy	1641
6.306.1.2	~DefaultPrefetchPolicy	1641
6.306.2	Member Function Documentation	1641
6.306.2.1	clone	1641
6.306.2.2	getDurableTopicPrefetch	1641
6.306.2.3	getMaxPrefetchLimit	1642
6.306.2.4	getQueueBrowserPrefetch	1642
6.306.2.5	getQueuePrefetch	1642

6.306.2.6	getTopicPrefetch . . . . .	1642
6.306.2.7	setDurableTopicPrefetch . . . . .	1643
6.306.2.8	setQueueBrowserPrefetch . . . . .	1643
6.306.2.9	setQueuePrefetch . . . . .	1643
6.306.2.10	setTopicPrefetch . . . . .	1643
6.306.3	Field Documentation . . . . .	1643
6.306.3.1	DEFAULT_DURABLE_TOPIC_PREFETCH . . . . .	1644
6.306.3.2	DEFAULT_QUEUE_BROWSER_PREFETCH . . . . .	1644
6.306.3.3	DEFAULT_QUEUE_PREFETCH . . . . .	1644
6.306.3.4	DEFAULT_TOPIC_PREFETCH . . . . .	1644
6.306.3.5	MAX_PREFETCH_SIZE . . . . .	1644
6.307	activemq::core::policies::DefaultRedeliveryPolicy Class Reference . . . . .	1644
6.307.1	Constructor & Destructor Documentation . . . . .	1645
6.307.1.1	DefaultRedeliveryPolicy . . . . .	1645
6.307.1.2	~DefaultRedeliveryPolicy . . . . .	1645
6.307.2	Member Function Documentation . . . . .	1645
6.307.2.1	clone . . . . .	1645
6.307.2.2	getBackOffMultiplier . . . . .	1645
6.307.2.3	getCollisionAvoidancePercent . . . . .	1645
6.307.2.4	getInitialRedeliveryDelay . . . . .	1646
6.307.2.5	getMaximumRedeliveries . . . . .	1646
6.307.2.6	getRedeliveryDelay . . . . .	1646
6.307.2.7	isUseCollisionAvoidance . . . . .	1646
6.307.2.8	isUseExponentialBackOff . . . . .	1647
6.307.2.9	setBackOffMultiplier . . . . .	1647
6.307.2.10	setCollisionAvoidancePercent . . . . .	1647
6.307.2.11	setInitialRedeliveryDelay . . . . .	1647
6.307.2.12	setMaximumRedeliveries . . . . .	1647
6.307.2.13	setUseCollisionAvoidance . . . . .	1648
6.307.2.14	setUseExponentialBackOff . . . . .	1648
6.308	decaf::internal::net::DefaultServerSocketFactory Class Reference . . . . .	1648
6.308.1	Detailed Description . . . . .	1649
6.308.2	Constructor & Destructor Documentation . . . . .	1650
6.308.2.1	DefaultServerSocketFactory . . . . .	1650



6.308.2.2 ~DefaultServerSocketFactory . . . . .	1650
6.308.3 Member Function Documentation . . . . .	1650
6.308.3.1 createServerSocket . . . . .	1650
6.308.3.2 createServerSocket . . . . .	1650
6.308.3.3 createServerSocket . . . . .	1651
6.308.3.4 createServerSocket . . . . .	1651
6.309decaf::internal::net::DefaultSocketFactory Class Reference . . . . .	1652
6.309.1 Detailed Description . . . . .	1654
6.309.2 Constructor & Destructor Documentation . . . . .	1654
6.309.2.1 DefaultSocketFactory . . . . .	1654
6.309.2.2 ~DefaultSocketFactory . . . . .	1654
6.309.3 Member Function Documentation . . . . .	1654
6.309.3.1 createSocket . . . . .	1654
6.309.3.2 createSocket . . . . .	1654
6.309.3.3 createSocket . . . . .	1655
6.309.3.4 createSocket . . . . .	1656
6.309.3.5 createSocket . . . . .	1656
6.310decaf::internal::net::ssl::DefaultSSLContext Class Reference . . . . .	1657
6.310.1 Detailed Description . . . . .	1657
6.310.2 Constructor & Destructor Documentation . . . . .	1657
6.310.2.1 DefaultSSLContext . . . . .	1657
6.310.2.2 ~DefaultSSLContext . . . . .	1658
6.310.3 Member Function Documentation . . . . .	1658
6.310.3.1 getContext . . . . .	1658
6.311decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference	1658
6.311.1 Detailed Description . . . . .	1660
6.311.2 Constructor & Destructor Documentation . . . . .	1660
6.311.2.1 DefaultSSLServerSocketFactory . . . . .	1660
6.311.2.2 ~DefaultSSLServerSocketFactory . . . . .	1660
6.311.3 Member Function Documentation . . . . .	1660
6.311.3.1 createServerSocket . . . . .	1660
6.311.3.2 createServerSocket . . . . .	1660
6.311.3.3 createServerSocket . . . . .	1661
6.311.3.4 createServerSocket . . . . .	1661

6.311.3.5	getDefaultCipherSuites . . . . .	1662
6.311.3.6	getSupportedCipherSuites . . . . .	1662
6.312	decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference . . .	1663
6.312.1	Detailed Description . . . . .	1666
6.312.2	Constructor & Destructor Documentation . . . . .	1666
6.312.2.1	DefaultSSLSocketFactory . . . . .	1666
6.312.2.2	~DefaultSSLSocketFactory . . . . .	1666
6.312.3	Member Function Documentation . . . . .	1666
6.312.3.1	createSocket . . . . .	1666
6.312.3.2	createSocket . . . . .	1666
6.312.3.3	createSocket . . . . .	1667
6.312.3.4	createSocket . . . . .	1668
6.312.3.5	createSocket . . . . .	1668
6.312.3.6	createSocket . . . . .	1669
6.312.3.7	getDefaultCipherSuites . . . . .	1669
6.312.3.8	getSupportedCipherSuites . . . . .	1670
6.313	activemq::transport::DefaultTransportListener Class Reference . . . . .	1670
6.313.1	Constructor & Destructor Documentation . . . . .	1671
6.313.1.1	~DefaultTransportListener . . . . .	1671
6.313.2	Member Function Documentation . . . . .	1671
6.313.2.1	onCommand . . . . .	1671
6.313.2.2	onException . . . . .	1671
6.313.2.3	transportInterrupted . . . . .	1671
6.313.2.4	transportResumed . . . . .	1671
6.314	decaf::util::zip::Deflater Class Reference . . . . .	1672
6.314.1	Detailed Description . . . . .	1674
6.314.2	Constructor & Destructor Documentation . . . . .	1674
6.314.2.1	Deflater . . . . .	1674
6.314.2.2	Deflater . . . . .	1674
6.314.2.3	~Deflater . . . . .	1674
6.314.3	Member Function Documentation . . . . .	1674
6.314.3.1	deflate . . . . .	1674
6.314.3.2	deflate . . . . .	1675
6.314.3.3	deflate . . . . .	1676

6.314.3.4 end . . . . .	1676
6.314.3.5 finish . . . . .	1676
6.314.3.6 finished . . . . .	1676
6.314.3.7 getAdler . . . . .	1676
6.314.3.8 getBytesRead . . . . .	1677
6.314.3.9 getBytesWritten . . . . .	1677
6.314.3.10needsInput . . . . .	1677
6.314.3.11reset . . . . .	1677
6.314.3.12setDictionary . . . . .	1677
6.314.3.13setDictionary . . . . .	1678
6.314.3.14setDictionary . . . . .	1678
6.314.3.15setInput . . . . .	1679
6.314.3.16setInput . . . . .	1679
6.314.3.17setInput . . . . .	1680
6.314.3.18setLevel . . . . .	1680
6.314.3.19setStrategy . . . . .	1680
6.314.4 Field Documentation . . . . .	1681
6.314.4.1 BEST_COMPRESSION . . . . .	1681
6.314.4.2 BEST_SPEED . . . . .	1681
6.314.4.3 DEFAULT_COMPRESSION . . . . .	1681
6.314.4.4 DEFAULT_STRATEGY . . . . .	1681
6.314.4.5 DEFLATED . . . . .	1681
6.314.4.6 FILTERED . . . . .	1681
6.314.4.7 HUFFMAN_ONLY . . . . .	1681
6.314.4.8 NO_COMPRESSION . . . . .	1681
6.315decaf::util::zip::DeflaterOutputStream Class Reference . . . . .	1682
6.315.1 Detailed Description . . . . .	1683
6.315.2 Constructor & Destructor Documentation . . . . .	1683
6.315.2.1 DeflaterOutputStream . . . . .	1683
6.315.2.2 DeflaterOutputStream . . . . .	1684
6.315.2.3 DeflaterOutputStream . . . . .	1684
6.315.2.4 ~DeflaterOutputStream . . . . .	1685
6.315.3 Member Function Documentation . . . . .	1685
6.315.3.1 close . . . . .	1685

6.315.3.2 deflate . . . . .	1685
6.315.3.3 doWriteArrayBounded . . . . .	1685
6.315.3.4 doWriteByte . . . . .	1685
6.315.3.5 finish . . . . .	1685
6.315.4 Field Documentation . . . . .	1686
6.315.4.1 buf . . . . .	1686
6.315.4.2 DEFAULT_BUFFER_SIZE . . . . .	1686
6.315.4.3 deflater . . . . .	1686
6.315.4.4 isDone . . . . .	1686
6.315.4.5 ownDeflater . . . . .	1686
6.316decaf::util::concurrent::Delayed Class Reference . . . . .	1686
6.316.1 Detailed Description . . . . .	1687
6.316.2 Constructor & Destructor Documentation . . . . .	1687
6.316.2.1 ~Delayed . . . . .	1687
6.316.3 Member Function Documentation . . . . .	1687
6.316.3.1 getDelay . . . . .	1687
6.317cms::DeliveryMode Class Reference . . . . .	1687
6.317.1 Detailed Description . . . . .	1688
6.317.2 Member Enumeration Documentation . . . . .	1688
6.317.2.1 DELIVERY_MODE . . . . .	1688
6.317.3 Constructor & Destructor Documentation . . . . .	1688
6.317.3.1 ~DeliveryMode . . . . .	1688
6.318cms::Destination Class Reference . . . . .	1688
6.318.1 Detailed Description . . . . .	1689
6.318.2 Member Enumeration Documentation . . . . .	1689
6.318.2.1 DestinationType . . . . .	1689
6.318.3 Constructor & Destructor Documentation . . . . .	1690
6.318.3.1 ~Destination . . . . .	1690
6.318.4 Member Function Documentation . . . . .	1690
6.318.4.1 clone . . . . .	1690
6.318.4.2 copy . . . . .	1690
6.318.4.3 getCMSProperties . . . . .	1690
6.318.4.4 getDestinationType . . . . .	1691

6.319activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference . . . . .	1691
6.319.1 Field Documentation . . . . .	1691
6.319.1.1 ANY_CHILD . . . . .	1691
6.319.1.2 ANY_DESCENDENT . . . . .	1691
6.320activemq::commands::DestinationInfo Class Reference . . . . .	1691
6.320.1 Constructor & Destructor Documentation . . . . .	1693
6.320.1.1 DestinationInfo . . . . .	1693
6.320.1.2 ~DestinationInfo . . . . .	1693
6.320.2 Member Function Documentation . . . . .	1693
6.320.2.1 cloneDataStructure . . . . .	1693
6.320.2.2 copyDataStructure . . . . .	1693
6.320.2.3 equals . . . . .	1693
6.320.2.4 getBrokerPath . . . . .	1694
6.320.2.5 getBrokerPath . . . . .	1694
6.320.2.6 getConnectionId . . . . .	1694
6.320.2.7 getConnectionId . . . . .	1694
6.320.2.8 getDataStructureType . . . . .	1694
6.320.2.9 getDestination . . . . .	1694
6.320.2.10getDestination . . . . .	1694
6.320.2.11getOperationType . . . . .	1694
6.320.2.12getTimeout . . . . .	1694
6.320.2.13setBrokerPath . . . . .	1694
6.320.2.14setConnectionId . . . . .	1695
6.320.2.15setDestination . . . . .	1695
6.320.2.16setOperationType . . . . .	1695
6.320.2.17setTimeout . . . . .	1695
6.320.2.18toString . . . . .	1695
6.320.2.19visit . . . . .	1695
6.320.3 Field Documentation . . . . .	1695
6.320.3.1 brokerPath . . . . .	1695
6.320.3.2 connectionId . . . . .	1695
6.320.3.3 destination . . . . .	1696
6.320.3.4 ID_DESTINATIONINFO . . . . .	1696

6.320.3.5 operationType . . . . .	1696
6.320.3.6 timeout . . . . .	1696
6.321activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference . . . . .	1696
6.321.1 Detailed Description . . . . .	1697
6.321.2 Constructor & Destructor Documentation . . . . .	1697
6.321.2.1 DestinationInfoMarshaller . . . . .	1697
6.321.2.2 ~DestinationInfoMarshaller . . . . .	1697
6.321.3 Member Function Documentation . . . . .	1697
6.321.3.1 createObject . . . . .	1697
6.321.3.2 getDataStructureType . . . . .	1697
6.321.3.3 looseMarshal . . . . .	1698
6.321.3.4 looseUnmarshal . . . . .	1698
6.321.3.5 tightMarshal1 . . . . .	1698
6.321.3.6 tightMarshal2 . . . . .	1699
6.321.3.7 tightUnmarshal . . . . .	1699
6.322activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference . . . . .	1700
6.322.1 Detailed Description . . . . .	1701
6.322.2 Constructor & Destructor Documentation . . . . .	1701
6.322.2.1 DestinationInfoMarshaller . . . . .	1701
6.322.2.2 ~DestinationInfoMarshaller . . . . .	1701
6.322.3 Member Function Documentation . . . . .	1701
6.322.3.1 createObject . . . . .	1701
6.322.3.2 getDataStructureType . . . . .	1701
6.322.3.3 looseMarshal . . . . .	1702
6.322.3.4 looseUnmarshal . . . . .	1702
6.322.3.5 tightMarshal1 . . . . .	1702
6.322.3.6 tightMarshal2 . . . . .	1703
6.322.3.7 tightUnmarshal . . . . .	1703
6.323activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference . . . . .	1704
6.323.1 Detailed Description . . . . .	1705
6.323.2 Constructor & Destructor Documentation . . . . .	1705
6.323.2.1 DestinationInfoMarshaller . . . . .	1705

6.323.2.2	~DestinationInfoMarshaller . . . . .	1705
6.323.3	Member Function Documentation . . . . .	1705
6.323.3.1	createObject . . . . .	1705
6.323.3.2	getDataStructureType . . . . .	1705
6.323.3.3	looseMarshal . . . . .	1706
6.323.3.4	looseUnmarshal . . . . .	1706
6.323.3.5	tightMarshal1 . . . . .	1706
6.323.3.6	tightMarshal2 . . . . .	1707
6.323.3.7	tightUnmarshal . . . . .	1707
6.324	activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference . . . . .	1708
6.324.1	Detailed Description . . . . .	1709
6.324.2	Constructor & Destructor Documentation . . . . .	1709
6.324.2.1	DestinationInfoMarshaller . . . . .	1709
6.324.2.2	~DestinationInfoMarshaller . . . . .	1709
6.324.3	Member Function Documentation . . . . .	1709
6.324.3.1	createObject . . . . .	1709
6.324.3.2	getDataStructureType . . . . .	1709
6.324.3.3	looseMarshal . . . . .	1710
6.324.3.4	looseUnmarshal . . . . .	1710
6.324.3.5	tightMarshal1 . . . . .	1710
6.324.3.6	tightMarshal2 . . . . .	1711
6.324.3.7	tightUnmarshal . . . . .	1711
6.325	activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller Class Reference . . . . .	1712
6.325.1	Detailed Description . . . . .	1713
6.325.2	Constructor & Destructor Documentation . . . . .	1713
6.325.2.1	DestinationInfoMarshaller . . . . .	1713
6.325.2.2	~DestinationInfoMarshaller . . . . .	1713
6.325.3	Member Function Documentation . . . . .	1713
6.325.3.1	createObject . . . . .	1713
6.325.3.2	getDataStructureType . . . . .	1713
6.325.3.3	looseMarshal . . . . .	1714
6.325.3.4	looseUnmarshal . . . . .	1714

6.325.3.5 tightMarshal1 . . . . .	1714
6.325.3.6 tightMarshal2 . . . . .	1715
6.325.3.7 tightUnmarshal . . . . .	1715
6.326activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference . . . . .	1716
6.326.1 Detailed Description . . . . .	1717
6.326.2 Constructor & Destructor Documentation . . . . .	1717
6.326.2.1 DestinationInfoMarshaller . . . . .	1717
6.326.2.2 ~DestinationInfoMarshaller . . . . .	1717
6.326.3 Member Function Documentation . . . . .	1717
6.326.3.1 createObject . . . . .	1717
6.326.3.2 getDataStructureType . . . . .	1717
6.326.3.3 looseMarshal . . . . .	1718
6.326.3.4 looseUnmarshal . . . . .	1718
6.326.3.5 tightMarshal1 . . . . .	1718
6.326.3.6 tightMarshal2 . . . . .	1719
6.326.3.7 tightUnmarshal . . . . .	1719
6.327activemq::cmsutil::DestinationResolver Class Reference . . . . .	1720
6.327.1 Detailed Description . . . . .	1720
6.327.2 Constructor & Destructor Documentation . . . . .	1720
6.327.2.1 ~DestinationResolver . . . . .	1721
6.327.3 Member Function Documentation . . . . .	1721
6.327.3.1 destroy . . . . .	1721
6.327.3.2 init . . . . .	1721
6.327.3.3 resolveDestinationName . . . . .	1721
6.328activemq::commands::DiscoveryEvent Class Reference . . . . .	1722
6.328.1 Constructor & Destructor Documentation . . . . .	1723
6.328.1.1 DiscoveryEvent . . . . .	1723
6.328.1.2 ~DiscoveryEvent . . . . .	1723
6.328.2 Member Function Documentation . . . . .	1723
6.328.2.1 cloneDataStructure . . . . .	1723
6.328.2.2 copyDataStructure . . . . .	1723
6.328.2.3 equals . . . . .	1723
6.328.2.4 getBrokerName . . . . .	1724



6.328.2.5	getBrokerName . . . . .	1724
6.328.2.6	getDataStructureType . . . . .	1724
6.328.2.7	getServiceName . . . . .	1724
6.328.2.8	getServiceName . . . . .	1724
6.328.2.9	setBrokerName . . . . .	1724
6.328.2.10	setServiceName . . . . .	1724
6.328.2.11	toString . . . . .	1724
6.328.3	Field Documentation . . . . .	1724
6.328.3.1	brokerName . . . . .	1724
6.328.3.2	ID_DISCOVERYEVENT . . . . .	1725
6.328.3.3	serviceName . . . . .	1725
6.329	activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller	
	Class Reference . . . . .	1725
6.329.1	Detailed Description . . . . .	1726
6.329.2	Constructor & Destructor Documentation . . . . .	1726
6.329.2.1	DiscoveryEventMarshaller . . . . .	1726
6.329.2.2	~DiscoveryEventMarshaller . . . . .	1726
6.329.3	Member Function Documentation . . . . .	1726
6.329.3.1	createObject . . . . .	1726
6.329.3.2	getDataStructureType . . . . .	1726
6.329.3.3	looseMarshal . . . . .	1727
6.329.3.4	looseUnmarshal . . . . .	1727
6.329.3.5	tightMarshal1 . . . . .	1727
6.329.3.6	tightMarshal2 . . . . .	1728
6.329.3.7	tightUnmarshal . . . . .	1728
6.330	activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	
	Class Reference . . . . .	1729
6.330.1	Detailed Description . . . . .	1730
6.330.2	Constructor & Destructor Documentation . . . . .	1730
6.330.2.1	DiscoveryEventMarshaller . . . . .	1730
6.330.2.2	~DiscoveryEventMarshaller . . . . .	1730
6.330.3	Member Function Documentation . . . . .	1730
6.330.3.1	createObject . . . . .	1730
6.330.3.2	getDataStructureType . . . . .	1730

6.330.3.3 looseMarshal . . . . .	1731
6.330.3.4 looseUnmarshal . . . . .	1731
6.330.3.5 tightMarshal1 . . . . .	1731
6.330.3.6 tightMarshal2 . . . . .	1732
6.330.3.7 tightUnmarshal . . . . .	1732
6.331activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	
Class Reference . . . . .	1733
6.331.1 Detailed Description . . . . .	1734
6.331.2 Constructor & Destructor Documentation . . . . .	1734
6.331.2.1 DiscoveryEventMarshaller . . . . .	1734
6.331.2.2 ~DiscoveryEventMarshaller . . . . .	1734
6.331.3 Member Function Documentation . . . . .	1734
6.331.3.1 createObject . . . . .	1734
6.331.3.2 getDataStructureType . . . . .	1734
6.331.3.3 looseMarshal . . . . .	1735
6.331.3.4 looseUnmarshal . . . . .	1735
6.331.3.5 tightMarshal1 . . . . .	1735
6.331.3.6 tightMarshal2 . . . . .	1736
6.331.3.7 tightUnmarshal . . . . .	1736
6.332activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	
Class Reference . . . . .	1737
6.332.1 Detailed Description . . . . .	1738
6.332.2 Constructor & Destructor Documentation . . . . .	1738
6.332.2.1 DiscoveryEventMarshaller . . . . .	1738
6.332.2.2 ~DiscoveryEventMarshaller . . . . .	1738
6.332.3 Member Function Documentation . . . . .	1738
6.332.3.1 createObject . . . . .	1738
6.332.3.2 getDataStructureType . . . . .	1738
6.332.3.3 looseMarshal . . . . .	1739
6.332.3.4 looseUnmarshal . . . . .	1739
6.332.3.5 tightMarshal1 . . . . .	1739
6.332.3.6 tightMarshal2 . . . . .	1740
6.332.3.7 tightUnmarshal . . . . .	1740
6.333activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	
Class Reference . . . . .	1741

6.333.1 Detailed Description . . . . .	1742
6.333.2 Constructor & Destructor Documentation . . . . .	1742
6.333.2.1 DiscoveryEventMarshaller . . . . .	1742
6.333.2.2 ~DiscoveryEventMarshaller . . . . .	1742
6.333.3 Member Function Documentation . . . . .	1742
6.333.3.1 createObject . . . . .	1742
6.333.3.2 getDataStructureType . . . . .	1742
6.333.3.3 looseMarshal . . . . .	1743
6.333.3.4 looseUnmarshal . . . . .	1743
6.333.3.5 tightMarshal1 . . . . .	1743
6.333.3.6 tightMarshal2 . . . . .	1744
6.333.3.7 tightUnmarshal . . . . .	1744
6.334activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller	
Class Reference . . . . .	1745
6.334.1 Detailed Description . . . . .	1746
6.334.2 Constructor & Destructor Documentation . . . . .	1746
6.334.2.1 DiscoveryEventMarshaller . . . . .	1746
6.334.2.2 ~DiscoveryEventMarshaller . . . . .	1746
6.334.3 Member Function Documentation . . . . .	1746
6.334.3.1 createObject . . . . .	1746
6.334.3.2 getDataStructureType . . . . .	1746
6.334.3.3 looseMarshal . . . . .	1747
6.334.3.4 looseUnmarshal . . . . .	1747
6.334.3.5 tightMarshal1 . . . . .	1747
6.334.3.6 tightMarshal2 . . . . .	1748
6.334.3.7 tightUnmarshal . . . . .	1748
6.335activemq::core::DispatchData Class Reference . . . . .	1749
6.335.1 Detailed Description . . . . .	1749
6.335.2 Constructor & Destructor Documentation . . . . .	1749
6.335.2.1 DispatchData . . . . .	1749
6.335.2.2 DispatchData . . . . .	1749
6.335.3 Member Function Documentation . . . . .	1749
6.335.3.1 getConsumerId . . . . .	1749
6.335.3.2 getMessage . . . . .	1749

6.336activemq::core::Dispatcher Class Reference . . . . .	1750
6.336.1 Detailed Description . . . . .	1750
6.336.2 Constructor & Destructor Documentation . . . . .	1750
6.336.2.1 ~Dispatcher . . . . .	1750
6.336.3 Member Function Documentation . . . . .	1750
6.336.3.1 dispatch . . . . .	1750
6.337decaf::lang::Double Class Reference . . . . .	1751
6.337.1 Constructor & Destructor Documentation . . . . .	1753
6.337.1.1 Double . . . . .	1753
6.337.1.2 Double . . . . .	1753
6.337.1.3 ~Double . . . . .	1753
6.337.2 Member Function Documentation . . . . .	1753
6.337.2.1 byteValue . . . . .	1753
6.337.2.2 compare . . . . .	1753
6.337.2.3 compareTo . . . . .	1754
6.337.2.4 compareTo . . . . .	1754
6.337.2.5 doubleToLongBits . . . . .	1754
6.337.2.6 doubleToRawLongBits . . . . .	1755
6.337.2.7 doubleValue . . . . .	1755
6.337.2.8 equals . . . . .	1756
6.337.2.9 equals . . . . .	1756
6.337.2.10floatValue . . . . .	1756
6.337.2.11intValue . . . . .	1756
6.337.2.12sInfinite . . . . .	1756
6.337.2.13sInfinite . . . . .	1757
6.337.2.14sNaN . . . . .	1757
6.337.2.15sNaN . . . . .	1757
6.337.2.16longBitsToDouble . . . . .	1757
6.337.2.17longValue . . . . .	1758
6.337.2.18operator< . . . . .	1758
6.337.2.19operator< . . . . .	1758
6.337.2.20operator== . . . . .	1758
6.337.2.21operator== . . . . .	1759
6.337.2.22parseDouble . . . . .	1759

6.337.2.23	shortValue . . . . .	1759
6.337.2.24	toHexString . . . . .	1760
6.337.2.25	toString . . . . .	1760
6.337.2.26	toString . . . . .	1760
6.337.2.27	valueOf . . . . .	1761
6.337.2.28	valueOf . . . . .	1761
6.337.3	Field Documentation . . . . .	1762
6.337.3.1	MAX_VALUE . . . . .	1762
6.337.3.2	MIN_VALUE . . . . .	1762
6.337.3.3	NaN . . . . .	1762
6.337.3.4	NEGATIVE_INFINITY . . . . .	1762
6.337.3.5	POSITIVE_INFINITY . . . . .	1762
6.337.3.6	SIZE . . . . .	1762
6.338	decaf::internal::nio::DoubleArrayBuffer Class Reference . . . . .	1762
6.338.1	Constructor & Destructor Documentation . . . . .	1766
6.338.1.1	DoubleArrayBuffer . . . . .	1766
6.338.1.2	DoubleArrayBuffer . . . . .	1767
6.338.1.3	DoubleArrayBuffer . . . . .	1767
6.338.1.4	DoubleArrayBuffer . . . . .	1768
6.338.1.5	~DoubleArrayBuffer . . . . .	1768
6.338.2	Member Function Documentation . . . . .	1768
6.338.2.1	array . . . . .	1768
6.338.2.2	arrayOffset . . . . .	1768
6.338.2.3	asReadOnlyBuffer . . . . .	1769
6.338.2.4	compact . . . . .	1769
6.338.2.5	duplicate . . . . .	1770
6.338.2.6	get . . . . .	1770
6.338.2.7	get . . . . .	1771
6.338.2.8	hasArray . . . . .	1771
6.338.2.9	isReadOnly . . . . .	1771
6.338.2.10	put . . . . .	1772
6.338.2.11	put . . . . .	1772
6.338.2.12	setReadOnly . . . . .	1773
6.338.2.13	slice . . . . .	1773

6.339decaf::nio::DoubleBuffer Class Reference . . . . .	1773
6.339.1 Detailed Description . . . . .	1775
6.339.2 Constructor & Destructor Documentation . . . . .	1776
6.339.2.1 DoubleBuffer . . . . .	1776
6.339.2.2 ~DoubleBuffer . . . . .	1776
6.339.3 Member Function Documentation . . . . .	1776
6.339.3.1 allocate . . . . .	1776
6.339.3.2 array . . . . .	1776
6.339.3.3 arrayOffset . . . . .	1777
6.339.3.4 asReadOnlyBuffer . . . . .	1777
6.339.3.5 compact . . . . .	1778
6.339.3.6 compareTo . . . . .	1778
6.339.3.7 duplicate . . . . .	1778
6.339.3.8 equals . . . . .	1779
6.339.3.9 get . . . . .	1779
6.339.3.10get . . . . .	1779
6.339.3.11get . . . . .	1780
6.339.3.12get . . . . .	1780
6.339.3.13hasArray . . . . .	1781
6.339.3.14operator< . . . . .	1781
6.339.3.15operator== . . . . .	1781
6.339.3.16put . . . . .	1781
6.339.3.17put . . . . .	1782
6.339.3.18put . . . . .	1782
6.339.3.19put . . . . .	1783
6.339.3.20put . . . . .	1784
6.339.3.21slice . . . . .	1784
6.339.3.22toString . . . . .	1785
6.339.3.23wrap . . . . .	1785
6.339.3.24wrap . . . . .	1785
6.340decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference . . . . .	1786
6.341activemq::cmsutil::DynamicDestinationResolver Class Reference . . . . .	1786
6.341.1 Detailed Description . . . . .	1787
6.341.2 Constructor & Destructor Documentation . . . . .	1787

6.341.2.1 DynamicDestinationResolver . . . . .	1787
6.341.2.2 DynamicDestinationResolver . . . . .	1787
6.341.2.3 ~DynamicDestinationResolver . . . . .	1787
6.341.3 Member Function Documentation . . . . .	1787
6.341.3.1 destroy . . . . .	1787
6.341.3.2 init . . . . .	1787
6.341.3.3 operator= . . . . .	1787
6.341.3.4 resolveDestinationName . . . . .	1788
6.342decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference . . . . .	1788
6.342.1 Constructor & Destructor Documentation . . . . .	1789
6.342.1.1 Entry . . . . .	1789
6.342.1.2 ~Entry . . . . .	1789
6.342.2 Member Function Documentation . . . . .	1789
6.342.2.1 getKey . . . . .	1789
6.342.2.2 getValue . . . . .	1789
6.342.2.3 setValue . . . . .	1789
6.343decaf::io::EOFException Class Reference . . . . .	1789
6.343.1 Constructor & Destructor Documentation . . . . .	1790
6.343.1.1 EOFException . . . . .	1790
6.343.1.2 EOFException . . . . .	1790
6.343.1.3 EOFException . . . . .	1790
6.343.1.4 EOFException . . . . .	1790
6.343.1.5 EOFException . . . . .	1791
6.343.1.6 EOFException . . . . .	1791
6.343.1.7 ~EOFException . . . . .	1791
6.343.2 Member Function Documentation . . . . .	1791
6.343.2.1 clone . . . . .	1791
6.344decaf::util::logging::ErrorManager Class Reference . . . . .	1792
6.344.1 Detailed Description . . . . .	1792
6.344.2 Constructor & Destructor Documentation . . . . .	1793
6.344.2.1 ErrorManager . . . . .	1793
6.344.2.2 ~ErrorManager . . . . .	1793
6.344.3 Member Function Documentation . . . . .	1793
6.344.3.1 error . . . . .	1793

6.344.4 Field Documentation . . . . .	1793
6.344.4.1 CLOSE_FAILURE . . . . .	1793
6.344.4.2 FLUSH_FAILURE . . . . .	1793
6.344.4.3 FORMAT_FAILURE . . . . .	1793
6.344.4.4 GENERIC_FAILURE . . . . .	1793
6.344.4.5 OPEN_FAILURE . . . . .	1794
6.344.4.6 WRITE_FAILURE . . . . .	1794
6.345decaf::lang::Exception Class Reference . . . . .	1794
6.345.1 Constructor & Destructor Documentation . . . . .	1795
6.345.1.1 Exception . . . . .	1795
6.345.1.2 Exception . . . . .	1796
6.345.1.3 Exception . . . . .	1796
6.345.1.4 Exception . . . . .	1796
6.345.1.5 Exception . . . . .	1796
6.345.1.6 ~Exception . . . . .	1797
6.345.2 Member Function Documentation . . . . .	1797
6.345.2.1 buildMessage . . . . .	1797
6.345.2.2 clone . . . . .	1797
6.345.2.3 getCause . . . . .	1798
6.345.2.4 getMessage . . . . .	1798
6.345.2.5 getStackTrace . . . . .	1798
6.345.2.6 getStackTraceString . . . . .	1798
6.345.2.7 initCause . . . . .	1799
6.345.2.8 operator= . . . . .	1799
6.345.2.9 printStackTrace . . . . .	1799
6.345.2.10printStackTrace . . . . .	1799
6.345.2.11setMark . . . . .	1799
6.345.2.12setMessage . . . . .	1800
6.345.2.13setStackTrace . . . . .	1800
6.345.2.14what . . . . .	1800
6.345.3 Field Documentation . . . . .	1800
6.345.3.1 cause . . . . .	1800
6.345.3.2 message . . . . .	1800
6.345.3.3 stackTrace . . . . .	1800



6.346cms::ExceptionListener Class Reference . . . . .	1801
6.346.1 Detailed Description . . . . .	1801
6.346.2 Constructor & Destructor Documentation . . . . .	1801
6.346.2.1 ~ExceptionListener . . . . .	1801
6.346.3 Member Function Documentation . . . . .	1801
6.346.3.1 onException . . . . .	1801
6.347activemq::commands::ExceptionResponse Class Reference . . . . .	1802
6.347.1 Constructor & Destructor Documentation . . . . .	1802
6.347.1.1 ExceptionResponse . . . . .	1802
6.347.1.2 ~ExceptionResponse . . . . .	1802
6.347.2 Member Function Documentation . . . . .	1803
6.347.2.1 cloneDataStructure . . . . .	1803
6.347.2.2 copyDataStructure . . . . .	1803
6.347.2.3 equals . . . . .	1803
6.347.2.4 getDataStructureType . . . . .	1803
6.347.2.5 getException . . . . .	1804
6.347.2.6 getException . . . . .	1804
6.347.2.7 setException . . . . .	1804
6.347.2.8 toString . . . . .	1804
6.347.3 Field Documentation . . . . .	1804
6.347.3.1 exception . . . . .	1804
6.347.3.2 ID_EXCEPTIONRESPONSE . . . . .	1804
6.348activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller Class Reference . . . . .	1804
6.348.1 Detailed Description . . . . .	1805
6.348.2 Constructor & Destructor Documentation . . . . .	1805
6.348.2.1 ExceptionResponseMarshaller . . . . .	1805
6.348.2.2 ~ExceptionResponseMarshaller . . . . .	1805
6.348.3 Member Function Documentation . . . . .	1805
6.348.3.1 createObject . . . . .	1806
6.348.3.2 getDataStructureType . . . . .	1806
6.348.3.3 looseMarshal . . . . .	1806
6.348.3.4 looseUnmarshal . . . . .	1807
6.348.3.5 tightMarshal1 . . . . .	1807

6.348.3.6	tightMarshal2 . . . . .	1808
6.348.3.7	tightUnmarshal . . . . .	1808
6.349	activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	
	Class Reference . . . . .	1809
6.349.1	Detailed Description . . . . .	1809
6.349.2	Constructor & Destructor Documentation . . . . .	1810
6.349.2.1	ExceptionResponseMarshaller . . . . .	1810
6.349.2.2	~ExceptionResponseMarshaller . . . . .	1810
6.349.3	Member Function Documentation . . . . .	1810
6.349.3.1	createObject . . . . .	1810
6.349.3.2	getDataStructureType . . . . .	1810
6.349.3.3	looseMarshal . . . . .	1810
6.349.3.4	looseUnmarshal . . . . .	1811
6.349.3.5	tightMarshal1 . . . . .	1811
6.349.3.6	tightMarshal2 . . . . .	1812
6.349.3.7	tightUnmarshal . . . . .	1812
6.350	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	
	Class Reference . . . . .	1813
6.350.1	Detailed Description . . . . .	1813
6.350.2	Constructor & Destructor Documentation . . . . .	1814
6.350.2.1	ExceptionResponseMarshaller . . . . .	1814
6.350.2.2	~ExceptionResponseMarshaller . . . . .	1814
6.350.3	Member Function Documentation . . . . .	1814
6.350.3.1	createObject . . . . .	1814
6.350.3.2	getDataStructureType . . . . .	1814
6.350.3.3	looseMarshal . . . . .	1814
6.350.3.4	looseUnmarshal . . . . .	1815
6.350.3.5	tightMarshal1 . . . . .	1815
6.350.3.6	tightMarshal2 . . . . .	1816
6.350.3.7	tightUnmarshal . . . . .	1816
6.351	activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	
	Class Reference . . . . .	1817
6.351.1	Detailed Description . . . . .	1817
6.351.2	Constructor & Destructor Documentation . . . . .	1818
6.351.2.1	ExceptionResponseMarshaller . . . . .	1818

6.351.2.2 ~ExceptionResponseMarshaller . . . . .	1818
6.351.3 Member Function Documentation . . . . .	1818
6.351.3.1 createObject . . . . .	1818
6.351.3.2 getDataStructureType . . . . .	1818
6.351.3.3 looseMarshal . . . . .	1818
6.351.3.4 looseUnmarshal . . . . .	1819
6.351.3.5 tightMarshal1 . . . . .	1819
6.351.3.6 tightMarshal2 . . . . .	1820
6.351.3.7 tightUnmarshal . . . . .	1820
6.352activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class Reference . . . . .	1821
6.352.1 Detailed Description . . . . .	1821
6.352.2 Constructor & Destructor Documentation . . . . .	1822
6.352.2.1 ExceptionResponseMarshaller . . . . .	1822
6.352.2.2 ~ExceptionResponseMarshaller . . . . .	1822
6.352.3 Member Function Documentation . . . . .	1822
6.352.3.1 createObject . . . . .	1822
6.352.3.2 getDataStructureType . . . . .	1822
6.352.3.3 looseMarshal . . . . .	1822
6.352.3.4 looseUnmarshal . . . . .	1823
6.352.3.5 tightMarshal1 . . . . .	1823
6.352.3.6 tightMarshal2 . . . . .	1824
6.352.3.7 tightUnmarshal . . . . .	1824
6.353activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class Reference . . . . .	1825
6.353.1 Detailed Description . . . . .	1825
6.353.2 Constructor & Destructor Documentation . . . . .	1826
6.353.2.1 ExceptionResponseMarshaller . . . . .	1826
6.353.2.2 ~ExceptionResponseMarshaller . . . . .	1826
6.353.3 Member Function Documentation . . . . .	1826
6.353.3.1 createObject . . . . .	1826
6.353.3.2 getDataStructureType . . . . .	1826
6.353.3.3 looseMarshal . . . . .	1826
6.353.3.4 looseUnmarshal . . . . .	1827

6.353.3.5 tightMarshal1 . . . . .	1827
6.353.3.6 tightMarshal2 . . . . .	1828
6.353.3.7 tightUnmarshal . . . . .	1828
6.354decaf::util::concurrent::ExecutionException Class Reference . . . . .	1829
6.354.1 Constructor & Destructor Documentation . . . . .	1829
6.354.1.1 ExecutionException . . . . .	1829
6.354.1.2 ExecutionException . . . . .	1829
6.354.1.3 ExecutionException . . . . .	1830
6.354.1.4 ExecutionException . . . . .	1830
6.354.1.5 ExecutionException . . . . .	1830
6.354.1.6 ExecutionException . . . . .	1830
6.354.1.7 ~ExecutionException . . . . .	1831
6.354.2 Member Function Documentation . . . . .	1831
6.354.2.1 clone . . . . .	1831
6.355decaf::util::concurrent::Executor Class Reference . . . . .	1831
6.355.1 Detailed Description . . . . .	1832
6.355.2 Constructor & Destructor Documentation . . . . .	1833
6.355.2.1 ~Executor . . . . .	1833
6.355.3 Member Function Documentation . . . . .	1833
6.355.3.1 execute . . . . .	1833
6.356decaf::util::concurrent::ExecutorService Class Reference . . . . .	1833
6.356.1 Detailed Description . . . . .	1834
6.356.2 Constructor & Destructor Documentation . . . . .	1834
6.356.2.1 ~ExecutorService . . . . .	1834
6.356.3 Member Function Documentation . . . . .	1834
6.356.3.1 awaitTermination . . . . .	1834
6.357activemq::transport::failover::FailoverTransport Class Reference . . . . .	1835
6.357.1 Constructor & Destructor Documentation . . . . .	1837
6.357.1.1 FailoverTransport . . . . .	1837
6.357.1.2 ~FailoverTransport . . . . .	1837
6.357.2 Member Function Documentation . . . . .	1837
6.357.2.1 add . . . . .	1838
6.357.2.2 addURI . . . . .	1838
6.357.2.3 close . . . . .	1838

6.357.2.4 getBackOffMultiplier . . . . .	1838
6.357.2.5 getBackupPoolSize . . . . .	1838
6.357.2.6 getInitialReconnectDelay . . . . .	1838
6.357.2.7 getMaxCacheSize . . . . .	1838
6.357.2.8 getMaxReconnectAttempts . . . . .	1839
6.357.2.9 getMaxReconnectDelay . . . . .	1839
6.357.2.10getReconnectDelay . . . . .	1839
6.357.2.11getRemoteAddress . . . . .	1839
6.357.2.12getStartupMaxReconnectAttempts . . . . .	1839
6.357.2.13getTimeout . . . . .	1839
6.357.2.14getTransportListener . . . . .	1839
6.357.2.15handleTransportFailure . . . . .	1839
6.357.2.16sBackup . . . . .	1840
6.357.2.17sClosed . . . . .	1840
6.357.2.18sConnected . . . . .	1840
6.357.2.19sFaultTolerant . . . . .	1840
6.357.2.20sInitialized . . . . .	1840
6.357.2.21sPending . . . . .	1841
6.357.2.22sRandomize . . . . .	1841
6.357.2.23sTrackMessages . . . . .	1841
6.357.2.24sTrackTransactionProducers . . . . .	1841
6.357.2.25sUseExponentialBackOff . . . . .	1841
6.357.2.26terate . . . . .	1841
6.357.2.27narrow . . . . .	1841
6.357.2.28oneway . . . . .	1842
6.357.2.29reconnect . . . . .	1842
6.357.2.30reconnect . . . . .	1842
6.357.2.31removeURI . . . . .	1842
6.357.2.32request . . . . .	1843
6.357.2.33request . . . . .	1843
6.357.2.34restoreTransport . . . . .	1844
6.357.2.35setBackOffMultiplier . . . . .	1844
6.357.2.36setBackup . . . . .	1844
6.357.2.37setBackupPoolSize . . . . .	1844

6.357.2.38	setConnectionInterruptProcessingComplete . . . . .	1844
6.357.2.39	setInitialized . . . . .	1844
6.357.2.40	setInitialReconnectDelay . . . . .	1844
6.357.2.41	setMaxCacheSize . . . . .	1845
6.357.2.42	setMaxReconnectAttempts . . . . .	1845
6.357.2.43	setMaxReconnectDelay . . . . .	1845
6.357.2.44	setRandomize . . . . .	1845
6.357.2.45	setReconnectDelay . . . . .	1845
6.357.2.46	setStartupMaxReconnectAttempts . . . . .	1845
6.357.2.47	setTimeout . . . . .	1845
6.357.2.48	setTrackMessages . . . . .	1845
6.357.2.49	setTrackTransactionProducers . . . . .	1845
6.357.2.50	setTransportListener . . . . .	1845
6.357.2.51	setUseExponentialBackOff . . . . .	1845
6.357.2.52	setWireFormat . . . . .	1845
6.357.2.53	start . . . . .	1846
6.357.2.54	stop . . . . .	1846
6.357.3	Friends And Related Function Documentation . . . . .	1846
6.357.3.1	FailoverTransportListener . . . . .	1846
6.358	activemq::transport::failover::FailoverTransportFactory Class Reference .	1846
6.358.1	Detailed Description . . . . .	1847
6.358.2	Constructor & Destructor Documentation . . . . .	1847
6.358.2.1	~FailoverTransportFactory . . . . .	1847
6.358.3	Member Function Documentation . . . . .	1847
6.358.3.1	create . . . . .	1847
6.358.3.2	createComposite . . . . .	1848
6.358.3.3	doCreateComposite . . . . .	1848
6.359	activemq::transport::failover::FailoverTransportListener Class Reference	1849
6.359.1	Detailed Description . . . . .	1849
6.359.2	Constructor & Destructor Documentation . . . . .	1849
6.359.2.1	FailoverTransportListener . . . . .	1849
6.359.2.2	~FailoverTransportListener . . . . .	1849
6.359.3	Member Function Documentation . . . . .	1849
6.359.3.1	onCommand . . . . .	1850

6.359.3.2 onException . . . . .	1850
6.359.3.3 transportInterrupted . . . . .	1850
6.359.3.4 transportResumed . . . . .	1850
6.360decaf::io::FileDescriptor Class Reference . . . . .	1850
6.360.1 Detailed Description . . . . .	1851
6.360.2 Constructor & Destructor Documentation . . . . .	1852
6.360.2.1 FileDescriptor . . . . .	1852
6.360.2.2 FileDescriptor . . . . .	1852
6.360.2.3 ~FileDescriptor . . . . .	1852
6.360.3 Member Function Documentation . . . . .	1852
6.360.3.1 sync . . . . .	1852
6.360.3.2 valid . . . . .	1852
6.360.4 Field Documentation . . . . .	1852
6.360.4.1 descriptor . . . . .	1852
6.360.4.2 err . . . . .	1852
6.360.4.3 in . . . . .	1852
6.360.4.4 out . . . . .	1853
6.360.4.5 readonly . . . . .	1853
6.361decaf::util::logging::Filter Class Reference . . . . .	1853
6.361.1 Detailed Description . . . . .	1853
6.361.2 Constructor & Destructor Documentation . . . . .	1853
6.361.2.1 ~Filter . . . . .	1853
6.361.3 Member Function Documentation . . . . .	1853
6.361.3.1 isLoggable . . . . .	1853
6.362decaf::io::FilterInputStream Class Reference . . . . .	1854
6.362.1 Detailed Description . . . . .	1856
6.362.2 Constructor & Destructor Documentation . . . . .	1856
6.362.2.1 FilterInputStream . . . . .	1856
6.362.2.2 ~FilterInputStream . . . . .	1857
6.362.3 Member Function Documentation . . . . .	1857
6.362.3.1 available . . . . .	1857
6.362.3.2 close . . . . .	1857
6.362.3.3 doReadArray . . . . .	1857
6.362.3.4 doReadArrayBounded . . . . .	1858

6.362.3.5 doReadByte . . . . .	1858
6.362.3.6 isClosed . . . . .	1858
6.362.3.7 mark . . . . .	1858
6.362.3.8 markSupported . . . . .	1859
6.362.3.9 reset . . . . .	1859
6.362.3.10 skip . . . . .	1860
6.362.4 Field Documentation . . . . .	1860
6.362.4.1 closed . . . . .	1860
6.362.4.2 inputStream . . . . .	1860
6.362.4.3 own . . . . .	1860
6.363decaf::io::FilterOutputStream Class Reference . . . . .	1861
6.363.1 Detailed Description . . . . .	1862
6.363.2 Constructor & Destructor Documentation . . . . .	1862
6.363.2.1 FilterOutputStream . . . . .	1862
6.363.2.2 ~FilterOutputStream . . . . .	1862
6.363.3 Member Function Documentation . . . . .	1862
6.363.3.1 close . . . . .	1863
6.363.3.2 doWriteArray . . . . .	1863
6.363.3.3 doWriteArrayBounded . . . . .	1863
6.363.3.4 doWriteByte . . . . .	1863
6.363.3.5 flush . . . . .	1864
6.363.3.6 isClosed . . . . .	1864
6.363.3.7 toString . . . . .	1864
6.363.4 Field Documentation . . . . .	1864
6.363.4.1 closed . . . . .	1864
6.363.4.2 outputStream . . . . .	1864
6.363.4.3 own . . . . .	1864
6.364decaf::lang::Float Class Reference . . . . .	1865
6.364.1 Constructor & Destructor Documentation . . . . .	1867
6.364.1.1 Float . . . . .	1867
6.364.1.2 Float . . . . .	1867
6.364.1.3 Float . . . . .	1867
6.364.1.4 ~Float . . . . .	1867
6.364.2 Member Function Documentation . . . . .	1867



6.364.2.1	byteValue	1867
6.364.2.2	compare	1867
6.364.2.3	compareTo	1868
6.364.2.4	compareTo	1868
6.364.2.5	doubleValue	1868
6.364.2.6	equals	1869
6.364.2.7	equals	1869
6.364.2.8	floatToIntBits	1869
6.364.2.9	floatToRawIntBits	1870
6.364.2.10	floatValue	1870
6.364.2.11	intBitsToFloat	1870
6.364.2.12	intValue	1871
6.364.2.13	isInfinite	1871
6.364.2.14	isInfinite	1871
6.364.2.15	sNaN	1871
6.364.2.16	sNaN	1871
6.364.2.17	longValue	1872
6.364.2.18	operator<	1872
6.364.2.19	operator<	1872
6.364.2.20	operator==	1873
6.364.2.21	operator==	1873
6.364.2.22	parseFloat	1873
6.364.2.23	shortValue	1874
6.364.2.24	toHexString	1874
6.364.2.25	toString	1874
6.364.2.26	toString	1875
6.364.2.27	valueOf	1875
6.364.2.28	valueOf	1876
6.364.3	Field Documentation	1876
6.364.3.1	MAX_VALUE	1876
6.364.3.2	MIN_VALUE	1876
6.364.3.3	NaN	1876
6.364.3.4	NEGATIVE_INFINITY	1876
6.364.3.5	POSITIVE_INFINITY	1876

6.364.3.6 SIZE . . . . .	1876
6.365decaf::internal::nio::FloatArrayBuffer Class Reference . . . . .	1876
6.365.1 Constructor & Destructor Documentation . . . . .	1880
6.365.1.1 FloatArrayBuffer . . . . .	1880
6.365.1.2 FloatArrayBuffer . . . . .	1881
6.365.1.3 FloatArrayBuffer . . . . .	1881
6.365.1.4 FloatArrayBuffer . . . . .	1882
6.365.1.5 ~FloatArrayBuffer . . . . .	1882
6.365.2 Member Function Documentation . . . . .	1882
6.365.2.1 array . . . . .	1882
6.365.2.2 arrayOffset . . . . .	1882
6.365.2.3 asReadOnlyBuffer . . . . .	1883
6.365.2.4 compact . . . . .	1883
6.365.2.5 duplicate . . . . .	1884
6.365.2.6 get . . . . .	1884
6.365.2.7 get . . . . .	1884
6.365.2.8 hasArray . . . . .	1885
6.365.2.9 isReadOnly . . . . .	1885
6.365.2.10put . . . . .	1885
6.365.2.11put . . . . .	1886
6.365.2.12setReadOnly . . . . .	1886
6.365.2.13slice . . . . .	1887
6.366decaf::nio::FloatBuffer Class Reference . . . . .	1887
6.366.1 Detailed Description . . . . .	1889
6.366.2 Constructor & Destructor Documentation . . . . .	1889
6.366.2.1 FloatBuffer . . . . .	1889
6.366.2.2 ~FloatBuffer . . . . .	1890
6.366.3 Member Function Documentation . . . . .	1890
6.366.3.1 allocate . . . . .	1890
6.366.3.2 array . . . . .	1890
6.366.3.3 arrayOffset . . . . .	1891
6.366.3.4 asReadOnlyBuffer . . . . .	1891
6.366.3.5 compact . . . . .	1892
6.366.3.6 compareTo . . . . .	1892

6.366.3.7 duplicate . . . . .	1892
6.366.3.8 equals . . . . .	1892
6.366.3.9 get . . . . .	1893
6.366.3.10get . . . . .	1893
6.366.3.11get . . . . .	1893
6.366.3.12get . . . . .	1894
6.366.3.13hasArray . . . . .	1894
6.366.3.14operator< . . . . .	1895
6.366.3.15operator== . . . . .	1895
6.366.3.16put . . . . .	1895
6.366.3.17put . . . . .	1895
6.366.3.18put . . . . .	1896
6.366.3.19put . . . . .	1897
6.366.3.20put . . . . .	1897
6.366.3.21slice . . . . .	1898
6.366.3.22toString . . . . .	1898
6.366.3.23wrap . . . . .	1898
6.366.3.24wrap . . . . .	1899
6.367decaf::io::Flushable Class Reference . . . . .	1899
6.367.1 Detailed Description . . . . .	1900
6.367.2 Constructor & Destructor Documentation . . . . .	1900
6.367.2.1 ~Flushable . . . . .	1900
6.367.3 Member Function Documentation . . . . .	1900
6.367.3.1 flush . . . . .	1900
6.368activemq::commands::FlushCommand Class Reference . . . . .	1900
6.368.1 Constructor & Destructor Documentation . . . . .	1901
6.368.1.1 FlushCommand . . . . .	1901
6.368.1.2 ~FlushCommand . . . . .	1901
6.368.2 Member Function Documentation . . . . .	1901
6.368.2.1 cloneDataStructure . . . . .	1901
6.368.2.2 copyDataStructure . . . . .	1902
6.368.2.3 equals . . . . .	1902
6.368.2.4 getDataStructureType . . . . .	1902
6.368.2.5 toString . . . . .	1902

6.368.2.6 visit . . . . .	1903
6.368.3 Field Documentation . . . . .	1903
6.368.3.1 ID_FLUSHCOMMAND . . . . .	1903
6.369activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller	
Class Reference . . . . .	1903
6.369.1 Detailed Description . . . . .	1904
6.369.2 Constructor & Destructor Documentation . . . . .	1904
6.369.2.1 FlushCommandMarshaller . . . . .	1904
6.369.2.2 ~FlushCommandMarshaller . . . . .	1904
6.369.3 Member Function Documentation . . . . .	1904
6.369.3.1 createObject . . . . .	1904
6.369.3.2 getDataStructureType . . . . .	1905
6.369.3.3 looseMarshal . . . . .	1905
6.369.3.4 looseUnmarshal . . . . .	1905
6.369.3.5 tightMarshal1 . . . . .	1906
6.369.3.6 tightMarshal2 . . . . .	1906
6.369.3.7 tightUnmarshal . . . . .	1907
6.370activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	
Class Reference . . . . .	1907
6.370.1 Detailed Description . . . . .	1908
6.370.2 Constructor & Destructor Documentation . . . . .	1908
6.370.2.1 FlushCommandMarshaller . . . . .	1908
6.370.2.2 ~FlushCommandMarshaller . . . . .	1908
6.370.3 Member Function Documentation . . . . .	1908
6.370.3.1 createObject . . . . .	1908
6.370.3.2 getDataStructureType . . . . .	1909
6.370.3.3 looseMarshal . . . . .	1909
6.370.3.4 looseUnmarshal . . . . .	1909
6.370.3.5 tightMarshal1 . . . . .	1910
6.370.3.6 tightMarshal2 . . . . .	1910
6.370.3.7 tightUnmarshal . . . . .	1911
6.371activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	
Class Reference . . . . .	1911
6.371.1 Detailed Description . . . . .	1912
6.371.2 Constructor & Destructor Documentation . . . . .	1912

6.371.2.1 FlushCommandMarshaller . . . . .	1912
6.371.2.2 ~FlushCommandMarshaller . . . . .	1912
6.371.3 Member Function Documentation . . . . .	1912
6.371.3.1 createObject . . . . .	1912
6.371.3.2 getDataStructureType . . . . .	1913
6.371.3.3 looseMarshal . . . . .	1913
6.371.3.4 looseUnmarshal . . . . .	1913
6.371.3.5 tightMarshal1 . . . . .	1914
6.371.3.6 tightMarshal2 . . . . .	1914
6.371.3.7 tightUnmarshal . . . . .	1915
6.372activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	
Class Reference . . . . .	1915
6.372.1 Detailed Description . . . . .	1916
6.372.2 Constructor & Destructor Documentation . . . . .	1916
6.372.2.1 FlushCommandMarshaller . . . . .	1916
6.372.2.2 ~FlushCommandMarshaller . . . . .	1916
6.372.3 Member Function Documentation . . . . .	1916
6.372.3.1 createObject . . . . .	1916
6.372.3.2 getDataStructureType . . . . .	1917
6.372.3.3 looseMarshal . . . . .	1917
6.372.3.4 looseUnmarshal . . . . .	1917
6.372.3.5 tightMarshal1 . . . . .	1918
6.372.3.6 tightMarshal2 . . . . .	1918
6.372.3.7 tightUnmarshal . . . . .	1919
6.373activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	
Class Reference . . . . .	1919
6.373.1 Detailed Description . . . . .	1920
6.373.2 Constructor & Destructor Documentation . . . . .	1920
6.373.2.1 FlushCommandMarshaller . . . . .	1920
6.373.2.2 ~FlushCommandMarshaller . . . . .	1920
6.373.3 Member Function Documentation . . . . .	1920
6.373.3.1 createObject . . . . .	1920
6.373.3.2 getDataStructureType . . . . .	1921
6.373.3.3 looseMarshal . . . . .	1921

6.373.3.4 looseUnmarshal . . . . .	1921
6.373.3.5 tightMarshal1 . . . . .	1922
6.373.3.6 tightMarshal2 . . . . .	1922
6.373.3.7 tightUnmarshal . . . . .	1923
6.374activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	
Class Reference . . . . .	1923
6.374.1 Detailed Description . . . . .	1924
6.374.2 Constructor & Destructor Documentation . . . . .	1924
6.374.2.1 FlushCommandMarshaller . . . . .	1924
6.374.2.2 ~FlushCommandMarshaller . . . . .	1924
6.374.3 Member Function Documentation . . . . .	1924
6.374.3.1 createObject . . . . .	1924
6.374.3.2 getDataStructureType . . . . .	1925
6.374.3.3 looseMarshal . . . . .	1925
6.374.3.4 looseUnmarshal . . . . .	1925
6.374.3.5 tightMarshal1 . . . . .	1926
6.374.3.6 tightMarshal2 . . . . .	1926
6.374.3.7 tightUnmarshal . . . . .	1927
6.375decaf::util::logging::Formatter Class Reference . . . . .	1927
6.375.1 Detailed Description . . . . .	1928
6.375.2 Constructor & Destructor Documentation . . . . .	1928
6.375.2.1 ~Formatter . . . . .	1928
6.375.3 Member Function Documentation . . . . .	1928
6.375.3.1 format . . . . .	1928
6.375.3.2 formatMessage . . . . .	1928
6.375.3.3 getHead . . . . .	1929
6.375.3.4 getTail . . . . .	1929
6.376decaf::util::concurrent::Future< V > Class Template Reference . . . . .	1929
6.376.1 Detailed Description . . . . .	1930
6.376.2 Constructor & Destructor Documentation . . . . .	1930
6.376.2.1 ~Future . . . . .	1930
6.376.3 Member Function Documentation . . . . .	1930
6.376.3.1 cancel . . . . .	1930
6.376.3.2 get . . . . .	1931

6.376.3.3 get . . . . .	1931
6.376.3.4 isCancelled . . . . .	1932
6.376.3.5 isDone . . . . .	1932
6.377activemq::transport::correlator::FutureResponse Class Reference . . . .	1932
6.377.1 Detailed Description . . . . .	1933
6.377.2 Constructor & Destructor Documentation . . . . .	1933
6.377.2.1 FutureResponse . . . . .	1933
6.377.2.2 ~FutureResponse . . . . .	1933
6.377.3 Member Function Documentation . . . . .	1933
6.377.3.1 getResponse . . . . .	1933
6.377.3.2 getResponse . . . . .	1933
6.377.3.3 getResponse . . . . .	1933
6.377.3.4 getResponse . . . . .	1934
6.377.3.5 setResponse . . . . .	1934
6.378decaf::security::GeneralSecurityException Class Reference . . . . .	1934
6.378.1 Constructor & Destructor Documentation . . . . .	1935
6.378.1.1 GeneralSecurityException . . . . .	1935
6.378.1.2 GeneralSecurityException . . . . .	1935
6.378.1.3 GeneralSecurityException . . . . .	1935
6.378.1.4 GeneralSecurityException . . . . .	1935
6.378.1.5 GeneralSecurityException . . . . .	1936
6.378.1.6 GeneralSecurityException . . . . .	1936
6.378.1.7 ~GeneralSecurityException . . . . .	1936
6.378.2 Member Function Documentation . . . . .	1936
6.378.2.1 clone . . . . .	1936
6.379decaf::internal::util::GenericResource< T > Class Template Reference .	1937
6.379.1 Detailed Description . . . . .	1937
6.379.2 Constructor & Destructor Documentation . . . . .	1938
6.379.2.1 GenericResource . . . . .	1938
6.379.2.2 ~GenericResource . . . . .	1938
6.379.3 Member Function Documentation . . . . .	1938
6.379.3.1 getManaged . . . . .	1938
6.379.3.2 setManaged . . . . .	1938
6.380gz_header_s Struct Reference . . . . .	1938

6.380.1 Field Documentation . . . . .	1938
6.380.1.1 comm_max . . . . .	1938
6.380.1.2 comment . . . . .	1939
6.380.1.3 done . . . . .	1939
6.380.1.4 extra . . . . .	1939
6.380.1.5 extra_len . . . . .	1939
6.380.1.6 extra_max . . . . .	1939
6.380.1.7 hcrc . . . . .	1939
6.380.1.8 name . . . . .	1939
6.380.1.9 name_max . . . . .	1939
6.380.1.10bs . . . . .	1939
6.380.1.11text . . . . .	1939
6.380.1.12time . . . . .	1939
6.380.1.13xflags . . . . .	1939
6.381gz_state Struct Reference . . . . .	1939
6.381.1 Field Documentation . . . . .	1940
6.381.1.1 direct . . . . .	1940
6.381.1.2 eof . . . . .	1940
6.381.1.3 err . . . . .	1940
6.381.1.4 fd . . . . .	1940
6.381.1.5 have . . . . .	1940
6.381.1.6 how . . . . .	1940
6.381.1.7 in . . . . .	1940
6.381.1.8 level . . . . .	1940
6.381.1.9 mode . . . . .	1940
6.381.1.10msg . . . . .	1940
6.381.1.11next . . . . .	1940
6.381.1.12but . . . . .	1940
6.381.1.13path . . . . .	1940
6.381.1.14pos . . . . .	1940
6.381.1.15raw . . . . .	1940
6.381.1.16seek . . . . .	1940
6.381.1.17size . . . . .	1940
6.381.1.18skip . . . . .	1941



6.381.1.19start . . . . .	1941
6.381.1.20strategy . . . . .	1941
6.381.1.21strm . . . . .	1941
6.381.1.22want . . . . .	1941
6.382decaf::util::logging::Handler Class Reference . . . . .	1941
6.382.1 Detailed Description . . . . .	1942
6.382.2 Constructor & Destructor Documentation . . . . .	1942
6.382.2.1 Handler . . . . .	1942
6.382.2.2 ~Handler . . . . .	1942
6.382.3 Member Function Documentation . . . . .	1942
6.382.3.1 flush . . . . .	1942
6.382.3.2 getErrorManager . . . . .	1942
6.382.3.3 getFilter . . . . .	1943
6.382.3.4 getFormatter . . . . .	1943
6.382.3.5 getLevel . . . . .	1943
6.382.3.6 isLoggable . . . . .	1943
6.382.3.7 publish . . . . .	1944
6.382.3.8 reportError . . . . .	1944
6.382.3.9 setErrorManager . . . . .	1944
6.382.3.10setFilter . . . . .	1944
6.382.3.11setFormatter . . . . .	1945
6.382.3.12setLevel . . . . .	1945
6.383decaf::internal::util::HexStringParser Class Reference . . . . .	1945
6.383.1 Constructor & Destructor Documentation . . . . .	1946
6.383.1.1 HexStringParser . . . . .	1946
6.383.1.2 ~HexStringParser . . . . .	1946
6.383.2 Member Function Documentation . . . . .	1946
6.383.2.1 parse . . . . .	1946
6.383.2.2 parseDouble . . . . .	1946
6.383.2.3 parseFloat . . . . .	1946
6.384activemq::wireformat::openwire::utils::HexTable Class Reference . . . . .	1947
6.384.1 Detailed Description . . . . .	1947
6.384.2 Constructor & Destructor Documentation . . . . .	1947
6.384.2.1 HexTable . . . . .	1947

6.384.2.2 ~HexTable . . . . .	1947
6.384.3 Member Function Documentation . . . . .	1947
6.384.3.1 operator[] . . . . .	1947
6.384.3.2 operator[] . . . . .	1948
6.384.3.3 size . . . . .	1948
6.385decaf::net::HttpRetryException Class Reference . . . . .	1948
6.385.1 Constructor & Destructor Documentation . . . . .	1949
6.385.1.1 HttpRetryException . . . . .	1949
6.385.1.2 HttpRetryException . . . . .	1949
6.385.1.3 HttpRetryException . . . . .	1949
6.385.1.4 HttpRetryException . . . . .	1949
6.385.1.5 HttpRetryException . . . . .	1950
6.385.1.6 HttpRetryException . . . . .	1950
6.385.1.7 ~HttpRetryException . . . . .	1950
6.385.2 Member Function Documentation . . . . .	1950
6.385.2.1 clone . . . . .	1950
6.386activemq::util::IdGenerator Class Reference . . . . .	1951
6.386.1 Constructor & Destructor Documentation . . . . .	1951
6.386.1.1 IdGenerator . . . . .	1951
6.386.1.2 IdGenerator . . . . .	1951
6.386.1.3 ~IdGenerator . . . . .	1951
6.386.2 Member Function Documentation . . . . .	1951
6.386.2.1 compare . . . . .	1952
6.386.2.2 generateId . . . . .	1952
6.386.2.3 getHostname . . . . .	1952
6.386.2.4 getSeedFromId . . . . .	1952
6.386.2.5 getSequenceFromId . . . . .	1952
6.387decaf::lang::exceptions::IllegalArgumentException Class Reference . . . . .	1953
6.387.1 Constructor & Destructor Documentation . . . . .	1953
6.387.1.1 IllegalArgumentException . . . . .	1953
6.387.1.2 IllegalArgumentException . . . . .	1953
6.387.1.3 IllegalArgumentException . . . . .	1954
6.387.1.4 IllegalArgumentException . . . . .	1954
6.387.1.5 IllegalArgumentException . . . . .	1954

6.387.1.6	IllegalArgumentException . . . . .	1954
6.387.1.7	~IllegalArgumentException . . . . .	1955
6.387.2	Member Function Documentation . . . . .	1955
6.387.2.1	clone . . . . .	1955
6.388	decaf::lang::exceptions::IllegalMonitorStateException Class Reference .	1955
6.388.1	Constructor & Destructor Documentation . . . . .	1956
6.388.1.1	IllegalMonitorStateException . . . . .	1956
6.388.1.2	IllegalMonitorStateException . . . . .	1956
6.388.1.3	IllegalMonitorStateException . . . . .	1956
6.388.1.4	IllegalMonitorStateException . . . . .	1956
6.388.1.5	IllegalMonitorStateException . . . . .	1957
6.388.1.6	IllegalMonitorStateException . . . . .	1957
6.388.1.7	~IllegalMonitorStateException . . . . .	1957
6.388.2	Member Function Documentation . . . . .	1957
6.388.2.1	clone . . . . .	1957
6.389	cms::IllegalStateException Class Reference . . . . .	1958
6.389.1	Detailed Description . . . . .	1958
6.389.2	Constructor & Destructor Documentation . . . . .	1958
6.389.2.1	IllegalStateException . . . . .	1958
6.389.2.2	IllegalStateException . . . . .	1959
6.389.2.3	IllegalStateException . . . . .	1959
6.389.2.4	IllegalStateException . . . . .	1959
6.389.2.5	~IllegalStateException . . . . .	1959
6.390	decaf::lang::exceptions::IllegalStateException Class Reference . . . . .	1959
6.390.1	Constructor & Destructor Documentation . . . . .	1960
6.390.1.1	IllegalStateException . . . . .	1960
6.390.1.2	IllegalStateException . . . . .	1960
6.390.1.3	IllegalStateException . . . . .	1960
6.390.1.4	IllegalStateException . . . . .	1960
6.390.1.5	IllegalStateException . . . . .	1960
6.390.1.6	IllegalStateException . . . . .	1961
6.390.1.7	~IllegalStateException . . . . .	1961
6.390.2	Member Function Documentation . . . . .	1961
6.390.2.1	clone . . . . .	1961

6.391	decaf::lang::exceptions::IllegalThreadStateException Class Reference	1962
6.391.1	Constructor & Destructor Documentation	1962
6.391.1.1	IllegalThreadStateException	1962
6.391.1.2	IllegalThreadStateException	1962
6.391.1.3	IllegalThreadStateException	1963
6.391.1.4	IllegalThreadStateException	1963
6.391.1.5	IllegalThreadStateException	1963
6.391.1.6	IllegalThreadStateException	1963
6.391.1.7	~IllegalThreadStateException	1964
6.391.2	Member Function Documentation	1964
6.391.2.1	clone	1964
6.392	activemq::transport::inactivity::InactivityMonitor Class Reference	1964
6.392.1	Constructor & Destructor Documentation	1965
6.392.1.1	InactivityMonitor	1965
6.392.1.2	InactivityMonitor	1965
6.392.1.3	~InactivityMonitor	1965
6.392.2	Member Function Documentation	1965
6.392.2.1	close	1965
6.392.2.2	getInitialDelayTime	1966
6.392.2.3	getReadCheckTime	1966
6.392.2.4	getWriteCheckTime	1966
6.392.2.5	isKeepAliveResponseRequired	1966
6.392.2.6	onCommand	1966
6.392.2.7	oneway	1966
6.392.2.8	onException	1967
6.392.2.9	setInitialDelayTime	1967
6.392.2.10	setKeepAliveResponseRequired	1967
6.392.2.11	setReadCheckTime	1967
6.392.2.12	setWriteCheckTime	1967
6.392.3	Friends And Related Function Documentation	1967
6.392.3.1	AsyncSignalReadErrorTask	1967
6.392.3.2	AsyncWriteTask	1967
6.392.3.3	ReadChecker	1967
6.392.3.4	WriteChecker	1967

6.393decaf::lang::exceptions::IndexOutOfBoundsException Class Reference . . . . .	1967
6.393.1 Constructor & Destructor Documentation . . . . .	1968
6.393.1.1 IndexOutOfBoundsException . . . . .	1968
6.393.1.2 IndexOutOfBoundsException . . . . .	1968
6.393.1.3 IndexOutOfBoundsException . . . . .	1968
6.393.1.4 IndexOutOfBoundsException . . . . .	1969
6.393.1.5 IndexOutOfBoundsException . . . . .	1969
6.393.1.6 IndexOutOfBoundsException . . . . .	1969
6.393.1.7 ~IndexOutOfBoundsException . . . . .	1969
6.393.2 Member Function Documentation . . . . .	1969
6.393.2.1 clone . . . . .	1970
6.394decaf::net::Inet4Address Class Reference . . . . .	1970
6.394.1 Constructor & Destructor Documentation . . . . .	1971
6.394.1.1 Inet4Address . . . . .	1971
6.394.1.2 Inet4Address . . . . .	1971
6.394.1.3 Inet4Address . . . . .	1971
6.394.1.4 ~Inet4Address . . . . .	1971
6.394.2 Member Function Documentation . . . . .	1971
6.394.2.1 isAnyLocalAddress . . . . .	1971
6.394.2.2 isLinkLocalAddress . . . . .	1971
6.394.2.3 isLoopbackAddress . . . . .	1972
6.394.2.4 isMCGlobal . . . . .	1972
6.394.2.5 isMCLinkLocal . . . . .	1972
6.394.2.6 isMCNodeLocal . . . . .	1972
6.394.2.7 isMCOrgLocal . . . . .	1972
6.394.2.8 isMCSiteLocal . . . . .	1973
6.394.2.9 isMulticastAddress . . . . .	1973
6.394.2.10 isSiteLocalAddress . . . . .	1973
6.394.3 Friends And Related Function Documentation . . . . .	1973
6.394.3.1 InetAddress . . . . .	1973
6.395decaf::net::Inet6Address Class Reference . . . . .	1973
6.395.1 Constructor & Destructor Documentation . . . . .	1974
6.395.1.1 Inet6Address . . . . .	1974
6.395.1.2 Inet6Address . . . . .	1974

6.395.1.3 Inet6Address . . . . .	1974
6.395.1.4 ~Inet6Address . . . . .	1974
6.395.2 Friends And Related Function Documentation . . . . .	1974
6.395.2.1 InetAddress . . . . .	1974
6.396decaf::net::InetAddress Class Reference . . . . .	1974
6.396.1 Detailed Description . . . . .	1976
6.396.2 Constructor & Destructor Documentation . . . . .	1976
6.396.2.1 InetAddress . . . . .	1976
6.396.2.2 InetAddress . . . . .	1976
6.396.2.3 InetAddress . . . . .	1976
6.396.2.4 ~InetAddress . . . . .	1977
6.396.3 Member Function Documentation . . . . .	1977
6.396.3.1 bytesToInt . . . . .	1977
6.396.3.2 getAddress . . . . .	1977
6.396.3.3 getAnyAddress . . . . .	1977
6.396.3.4 getByAddress . . . . .	1977
6.396.3.5 getByAddress . . . . .	1978
6.396.3.6 getHostAddress . . . . .	1978
6.396.3.7 getHostName . . . . .	1978
6.396.3.8 getLocalHost . . . . .	1979
6.396.3.9 getLoopbackAddress . . . . .	1979
6.396.3.10sAnyLocalAddress . . . . .	1979
6.396.3.11isLinkLocalAddress . . . . .	1979
6.396.3.12sLoopbackAddress . . . . .	1979
6.396.3.13sMCGlobal . . . . .	1980
6.396.3.14sMCLinkLocal . . . . .	1980
6.396.3.15sMCNodeLocal . . . . .	1980
6.396.3.16sMCOrgLocal . . . . .	1980
6.396.3.17sMCSiteLocal . . . . .	1981
6.396.3.18sMulticastAddress . . . . .	1981
6.396.3.19sSiteLocalAddress . . . . .	1981
6.396.3.20toString . . . . .	1981
6.396.4 Field Documentation . . . . .	1981
6.396.4.1 addressBytes . . . . .	1981

6.396.4.2 anyBytes . . . . .	1982
6.396.4.3 hostname . . . . .	1982
6.396.4.4 loopbackBytes . . . . .	1982
6.396.4.5 reached . . . . .	1982
6.397decaf::net::InetSocketAddress Class Reference . . . . .	1982
6.397.1 Constructor & Destructor Documentation . . . . .	1982
6.397.1.1 InetSocketAddress . . . . .	1982
6.397.1.2 ~InetSocketAddress . . . . .	1982
6.398inflate_state Struct Reference . . . . .	1982
6.398.1 Field Documentation . . . . .	1983
6.398.1.1 back . . . . .	1983
6.398.1.2 bits . . . . .	1983
6.398.1.3 check . . . . .	1983
6.398.1.4 codes . . . . .	1983
6.398.1.5 distbits . . . . .	1984
6.398.1.6 distcode . . . . .	1984
6.398.1.7 dmax . . . . .	1984
6.398.1.8 extra . . . . .	1984
6.398.1.9 flags . . . . .	1984
6.398.1.10have . . . . .	1984
6.398.1.11havedict . . . . .	1984
6.398.1.12head . . . . .	1984
6.398.1.13hold . . . . .	1984
6.398.1.14last . . . . .	1984
6.398.1.15enbits . . . . .	1984
6.398.1.16encode . . . . .	1984
6.398.1.17length . . . . .	1984
6.398.1.18ens . . . . .	1984
6.398.1.19mode . . . . .	1984
6.398.1.20ncode . . . . .	1984
6.398.1.21ndist . . . . .	1984
6.398.1.22next . . . . .	1984
6.398.1.23nlen . . . . .	1984
6.398.1.24offset . . . . .	1984

6.398.1.25sane . . . . .	1984
6.398.1.26total . . . . .	1984
6.398.1.27was . . . . .	1984
6.398.1.28wbits . . . . .	1984
6.398.1.29whave . . . . .	1985
6.398.1.30window . . . . .	1985
6.398.1.31wnext . . . . .	1985
6.398.1.32work . . . . .	1985
6.398.1.33wrap . . . . .	1985
6.398.1.34wsize . . . . .	1985
6.399decaf::util::zip::Inflater Class Reference . . . . .	1985
6.399.1 Detailed Description . . . . .	1986
6.399.2 Constructor & Destructor Documentation . . . . .	1987
6.399.2.1 Inflater . . . . .	1987
6.399.2.2 Inflater . . . . .	1987
6.399.2.3 ~Inflater . . . . .	1987
6.399.3 Member Function Documentation . . . . .	1987
6.399.3.1 end . . . . .	1987
6.399.3.2 finish . . . . .	1988
6.399.3.3 finished . . . . .	1988
6.399.3.4 getAdler . . . . .	1988
6.399.3.5 getBytesRead . . . . .	1988
6.399.3.6 getBytesWritten . . . . .	1988
6.399.3.7 getRemaining . . . . .	1989
6.399.3.8 inflate . . . . .	1989
6.399.3.9 inflate . . . . .	1989
6.399.3.10inflate . . . . .	1990
6.399.3.11needsDictionary . . . . .	1990
6.399.3.12needsInput . . . . .	1991
6.399.3.13reset . . . . .	1991
6.399.3.14setDictionary . . . . .	1991
6.399.3.15setDictionary . . . . .	1992
6.399.3.16setDictionary . . . . .	1992
6.399.3.17setInput . . . . .	1993



6.399.3.18	setInput	1993
6.399.3.19	setInput	1994
6.400	decaf::util::zip::InflaterInputStream Class Reference	1994
6.400.1	Detailed Description	1997
6.400.2	Constructor & Destructor Documentation	1997
6.400.2.1	InflaterInputStream	1997
6.400.2.2	InflaterInputStream	1997
6.400.2.3	InflaterInputStream	1997
6.400.2.4	~InflaterInputStream	1998
6.400.3	Member Function Documentation	1998
6.400.3.1	available	1998
6.400.3.2	close	1998
6.400.3.3	doReadArrayBounded	1999
6.400.3.4	doReadByte	1999
6.400.3.5	fill	1999
6.400.3.6	mark	1999
6.400.3.7	markSupported	2000
6.400.3.8	reset	2000
6.400.3.9	skip	2001
6.400.4	Field Documentation	2001
6.400.4.1	atEOF	2001
6.400.4.2	buff	2001
6.400.4.3	DEFAULT_BUFFER_SIZE	2001
6.400.4.4	inflater	2001
6.400.4.5	length	2002
6.400.4.6	ownInflater	2002
6.401	decaf::io::InputStream Class Reference	2002
6.401.1	Detailed Description	2004
6.401.2	Constructor & Destructor Documentation	2004
6.401.2.1	InputStream	2004
6.401.2.2	~InputStream	2004
6.401.3	Member Function Documentation	2004
6.401.3.1	available	2004
6.401.3.2	close	2004

6.401.3.3 doReadArray . . . . .	2005
6.401.3.4 doReadArrayBounded . . . . .	2005
6.401.3.5 doReadByte . . . . .	2005
6.401.3.6 lock . . . . .	2005
6.401.3.7 mark . . . . .	2006
6.401.3.8 markSupported . . . . .	2006
6.401.3.9 notify . . . . .	2006
6.401.3.10 notifyAll . . . . .	2007
6.401.3.11 read . . . . .	2007
6.401.3.12 read . . . . .	2008
6.401.3.13 read . . . . .	2009
6.401.3.14 reset . . . . .	2009
6.401.3.15 skip . . . . .	2010
6.401.3.16 toString . . . . .	2011
6.401.3.17 tryLock . . . . .	2011
6.401.3.18 unlock . . . . .	2011
6.401.3.19 wait . . . . .	2011
6.401.3.20 wait . . . . .	2012
6.401.3.21 wait . . . . .	2012
6.402 decaf::io::InputStreamReader Class Reference . . . . .	2013
6.402.1 Detailed Description . . . . .	2014
6.402.2 Constructor & Destructor Documentation . . . . .	2014
6.402.2.1 InputStreamReader . . . . .	2014
6.402.2.2 ~InputStreamReader . . . . .	2014
6.402.3 Member Function Documentation . . . . .	2014
6.402.3.1 checkClosed . . . . .	2014
6.402.3.2 close . . . . .	2014
6.402.3.3 doReadArrayBounded . . . . .	2015
6.402.3.4 ready . . . . .	2015
6.403 decaf::internal::nio::IntArrayBuffer Class Reference . . . . .	2015
6.403.1 Constructor & Destructor Documentation . . . . .	2019
6.403.1.1 IntArrayBuffer . . . . .	2019
6.403.1.2 IntArrayBuffer . . . . .	2020
6.403.1.3 IntArrayBuffer . . . . .	2020

6.403.1.4 IntArrayBuffer . . . . .	2021
6.403.1.5 ~IntArrayBuffer . . . . .	2021
6.403.2 Member Function Documentation . . . . .	2021
6.403.2.1 array . . . . .	2021
6.403.2.2 arrayOffset . . . . .	2021
6.403.2.3 asReadOnlyBuffer . . . . .	2022
6.403.2.4 compact . . . . .	2022
6.403.2.5 duplicate . . . . .	2023
6.403.2.6 get . . . . .	2023
6.403.2.7 get . . . . .	2023
6.403.2.8 hasArray . . . . .	2024
6.403.2.9 isReadOnly . . . . .	2024
6.403.2.10put . . . . .	2024
6.403.2.11put . . . . .	2025
6.403.2.12setReadOnly . . . . .	2025
6.403.2.13slice . . . . .	2026
6.404decaf::nio::IntBuffer Class Reference . . . . .	2026
6.404.1 Detailed Description . . . . .	2028
6.404.2 Constructor & Destructor Documentation . . . . .	2028
6.404.2.1 IntBuffer . . . . .	2028
6.404.2.2 ~IntBuffer . . . . .	2029
6.404.3 Member Function Documentation . . . . .	2029
6.404.3.1 allocate . . . . .	2029
6.404.3.2 array . . . . .	2029
6.404.3.3 arrayOffset . . . . .	2030
6.404.3.4 asReadOnlyBuffer . . . . .	2030
6.404.3.5 compact . . . . .	2030
6.404.3.6 compareTo . . . . .	2031
6.404.3.7 duplicate . . . . .	2031
6.404.3.8 equals . . . . .	2031
6.404.3.9 get . . . . .	2031
6.404.3.10get . . . . .	2032
6.404.3.11get . . . . .	2032
6.404.3.12get . . . . .	2033

6.404.3.13	hasArray . . . . .	2033
6.404.3.14	operator< . . . . .	2034
6.404.3.15	operator== . . . . .	2034
6.404.3.16	put . . . . .	2034
6.404.3.17	put . . . . .	2034
6.404.3.18	put . . . . .	2035
6.404.3.19	put . . . . .	2036
6.404.3.20	put . . . . .	2036
6.404.3.21	slice . . . . .	2037
6.404.3.22	toString . . . . .	2037
6.404.3.23	wrap . . . . .	2037
6.404.3.24	wrap . . . . .	2038
6.405	decaf::lang::Integer Class Reference . . . . .	2038
6.405.1	Constructor & Destructor Documentation . . . . .	2041
6.405.1.1	Integer . . . . .	2041
6.405.1.2	Integer . . . . .	2041
6.405.1.3	~Integer . . . . .	2041
6.405.2	Member Function Documentation . . . . .	2041
6.405.2.1	bitCount . . . . .	2041
6.405.2.2	byteValue . . . . .	2042
6.405.2.3	compareTo . . . . .	2042
6.405.2.4	compareTo . . . . .	2042
6.405.2.5	decode . . . . .	2042
6.405.2.6	doubleValue . . . . .	2043
6.405.2.7	equals . . . . .	2043
6.405.2.8	equals . . . . .	2043
6.405.2.9	floatValue . . . . .	2044
6.405.2.10	highestOneBit . . . . .	2044
6.405.2.11	intValue . . . . .	2044
6.405.2.12	longValue . . . . .	2044
6.405.2.13	lowestOneBit . . . . .	2045
6.405.2.14	numberOfLeadingZeros . . . . .	2045
6.405.2.15	numberOfTrailingZeros . . . . .	2045
6.405.2.16	operator< . . . . .	2046

6.405.2.17	operator<	2046
6.405.2.18	operator==	2047
6.405.2.19	operator==	2047
6.405.2.20	parseInt	2047
6.405.2.21	parseInt	2048
6.405.2.22	reverse	2048
6.405.2.23	reverseBytes	2048
6.405.2.24	rotateLeft	2049
6.405.2.25	rotateRight	2049
6.405.2.26	shortValue	2050
6.405.2.27	signum	2050
6.405.2.28	toBinaryString	2050
6.405.2.29	toHexString	2051
6.405.2.30	toOctalString	2051
6.405.2.31	toString	2051
6.405.2.32	toString	2052
6.405.2.33	toString	2052
6.405.2.34	valueOf	2052
6.405.2.35	valueOf	2053
6.405.2.36	valueOf	2053
6.405.3	Field Documentation	2054
6.405.3.1	MAX_VALUE	2054
6.405.3.2	MIN_VALUE	2054
6.405.3.3	SIZE	2054
6.406	activemq::commands::IntegerResponse Class Reference	2054
6.406.1	Constructor & Destructor Documentation	2055
6.406.1.1	IntegerResponse	2055
6.406.1.2	~IntegerResponse	2055
6.406.2	Member Function Documentation	2055
6.406.2.1	cloneDataStructure	2055
6.406.2.2	copyDataStructure	2055
6.406.2.3	equals	2055
6.406.2.4	getDataStructureType	2056
6.406.2.5	getResult	2056

6.406.2.6 setResult . . . . .	2056
6.406.2.7 toString . . . . .	2056
6.406.3 Field Documentation . . . . .	2056
6.406.3.1 ID_INTEGERRESPONSE . . . . .	2056
6.406.3.2 result . . . . .	2056
6.407activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller	
Class Reference . . . . .	2057
6.407.1 Detailed Description . . . . .	2057
6.407.2 Constructor & Destructor Documentation . . . . .	2058
6.407.2.1 IntegerResponseMarshaller . . . . .	2058
6.407.2.2 ~IntegerResponseMarshaller . . . . .	2058
6.407.3 Member Function Documentation . . . . .	2058
6.407.3.1 createObject . . . . .	2058
6.407.3.2 getDataStructureType . . . . .	2058
6.407.3.3 looseMarshal . . . . .	2058
6.407.3.4 looseUnmarshal . . . . .	2059
6.407.3.5 tightMarshal1 . . . . .	2059
6.407.3.6 tightMarshal2 . . . . .	2060
6.407.3.7 tightUnmarshal . . . . .	2060
6.408activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	
Class Reference . . . . .	2061
6.408.1 Detailed Description . . . . .	2061
6.408.2 Constructor & Destructor Documentation . . . . .	2062
6.408.2.1 IntegerResponseMarshaller . . . . .	2062
6.408.2.2 ~IntegerResponseMarshaller . . . . .	2062
6.408.3 Member Function Documentation . . . . .	2062
6.408.3.1 createObject . . . . .	2062
6.408.3.2 getDataStructureType . . . . .	2062
6.408.3.3 looseMarshal . . . . .	2062
6.408.3.4 looseUnmarshal . . . . .	2063
6.408.3.5 tightMarshal1 . . . . .	2063
6.408.3.6 tightMarshal2 . . . . .	2064
6.408.3.7 tightUnmarshal . . . . .	2064
6.409activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	
Class Reference . . . . .	2065

6.409.1 Detailed Description . . . . .	2065
6.409.2 Constructor & Destructor Documentation . . . . .	2066
6.409.2.1 IntegerResponseMarshaller . . . . .	2066
6.409.2.2 ~IntegerResponseMarshaller . . . . .	2066
6.409.3 Member Function Documentation . . . . .	2066
6.409.3.1 createObject . . . . .	2066
6.409.3.2 getDataStructureType . . . . .	2066
6.409.3.3 looseMarshal . . . . .	2066
6.409.3.4 looseUnmarshal . . . . .	2067
6.409.3.5 tightMarshal1 . . . . .	2067
6.409.3.6 tightMarshal2 . . . . .	2068
6.409.3.7 tightUnmarshal . . . . .	2068
6.410activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	
Class Reference . . . . .	2069
6.410.1 Detailed Description . . . . .	2069
6.410.2 Constructor & Destructor Documentation . . . . .	2070
6.410.2.1 IntegerResponseMarshaller . . . . .	2070
6.410.2.2 ~IntegerResponseMarshaller . . . . .	2070
6.410.3 Member Function Documentation . . . . .	2070
6.410.3.1 createObject . . . . .	2070
6.410.3.2 getDataStructureType . . . . .	2070
6.410.3.3 looseMarshal . . . . .	2070
6.410.3.4 looseUnmarshal . . . . .	2071
6.410.3.5 tightMarshal1 . . . . .	2071
6.410.3.6 tightMarshal2 . . . . .	2072
6.410.3.7 tightUnmarshal . . . . .	2072
6.411activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	
Class Reference . . . . .	2073
6.411.1 Detailed Description . . . . .	2073
6.411.2 Constructor & Destructor Documentation . . . . .	2074
6.411.2.1 IntegerResponseMarshaller . . . . .	2074
6.411.2.2 ~IntegerResponseMarshaller . . . . .	2074
6.411.3 Member Function Documentation . . . . .	2074
6.411.3.1 createObject . . . . .	2074

6.411.3.2	getDataStructureType . . . . .	2074
6.411.3.3	looseMarshal . . . . .	2074
6.411.3.4	looseUnmarshal . . . . .	2075
6.411.3.5	tightMarshal1 . . . . .	2075
6.411.3.6	tightMarshal2 . . . . .	2076
6.411.3.7	tightUnmarshal . . . . .	2076
6.412	activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller	
	Class Reference . . . . .	2077
6.412.1	Detailed Description . . . . .	2077
6.412.2	Constructor & Destructor Documentation . . . . .	2078
6.412.2.1	IntegerResponseMarshaller . . . . .	2078
6.412.2.2	~IntegerResponseMarshaller . . . . .	2078
6.412.3	Member Function Documentation . . . . .	2078
6.412.3.1	createObject . . . . .	2078
6.412.3.2	getDataStructureType . . . . .	2078
6.412.3.3	looseMarshal . . . . .	2078
6.412.3.4	looseUnmarshal . . . . .	2079
6.412.3.5	tightMarshal1 . . . . .	2079
6.412.3.6	tightMarshal2 . . . . .	2080
6.412.3.7	tightUnmarshal . . . . .	2080
6.413	internal_state Struct Reference . . . . .	2081
6.413.1	Field Documentation . . . . .	2082
6.413.1.1	bi_buf . . . . .	2082
6.413.1.2	bi_valid . . . . .	2082
6.413.1.3	bl_count . . . . .	2082
6.413.1.4	bl_desc . . . . .	2082
6.413.1.5	bl_tree . . . . .	2082
6.413.1.6	block_start . . . . .	2082
6.413.1.7	d_buf . . . . .	2082
6.413.1.8	d_desc . . . . .	2082
6.413.1.9	depth . . . . .	2082
6.413.1.10	dummy . . . . .	2082
6.413.1.11	dyn_dtree . . . . .	2082
6.413.1.12	dyn_ltree . . . . .	2082



6.413.1.13	good_match	2083
6.413.1.14	gzhead	2083
6.413.1.15	gzindex	2083
6.413.1.16	hash_bits	2083
6.413.1.17	hash_mask	2083
6.413.1.18	hash_shift	2083
6.413.1.19	hash_size	2083
6.413.1.20	head	2083
6.413.1.21	heap	2083
6.413.1.22	heap_len	2083
6.413.1.23	heap_max	2083
6.413.1.24	high_water	2083
6.413.1.25	ins_h	2083
6.413.1.26	buf	2083
6.413.1.27	desc	2083
6.413.1.28	last_eob_len	2083
6.413.1.29	last_flush	2083
6.413.1.30	last_lit	2083
6.413.1.31	level	2083
6.413.1.32	lit_bufsize	2083
6.413.1.33	lookahead	2083
6.413.1.34	match_available	2083
6.413.1.35	match_length	2083
6.413.1.36	match_start	2083
6.413.1.37	matches	2084
6.413.1.38	max_chain_length	2084
6.413.1.39	max_lazy_match	2084
6.413.1.40	method	2084
6.413.1.41	nice_match	2084
6.413.1.42	opt_len	2084
6.413.1.43	pending	2084
6.413.1.44	pending_buf	2084
6.413.1.45	pending_buf_size	2084
6.413.1.46	pending_out	2084

6.413.1.47	prev	2084
6.413.1.48	prev_length	2084
6.413.1.49	prev_match	2084
6.413.1.50	static_len	2084
6.413.1.51	status	2084
6.413.1.52	strategy	2084
6.413.1.53	strm	2084
6.413.1.54	strstart	2084
6.413.1.55	w_bits	2084
6.413.1.56	w_mask	2084
6.413.1.57	w_size	2084
6.413.1.58	window	2084
6.413.1.59	window_size	2084
6.413.1.60	wrap	2085
6.414	activemq::transport::mock::InternalCommandListener Class Reference	2085
6.414.1	Detailed Description	2085
6.414.2	Constructor & Destructor Documentation	2085
6.414.2.1	InternalCommandListener	2085
6.414.2.2	~InternalCommandListener	2085
6.414.3	Member Function Documentation	2086
6.414.3.1	onCommand	2086
6.414.3.2	run	2086
6.414.3.3	setResponseBuilder	2086
6.414.3.4	setTransport	2086
6.415	decaf::lang::exceptions::InterruptedException Class Reference	2086
6.415.1	Constructor & Destructor Documentation	2087
6.415.1.1	InterruptedException	2087
6.415.1.2	InterruptedException	2087
6.415.1.3	InterruptedException	2087
6.415.1.4	InterruptedException	2088
6.415.1.5	InterruptedException	2088
6.415.1.6	InterruptedException	2088
6.415.1.7	~InterruptedException	2088
6.415.2	Member Function Documentation	2088

6.415.2.1 clone . . . . .	2089
6.416decaf::io::InterruptedIOException Class Reference . . . . .	2089
6.416.1 Constructor & Destructor Documentation . . . . .	2090
6.416.1.1 InterruptedIOException . . . . .	2090
6.416.1.2 InterruptedIOException . . . . .	2090
6.416.1.3 InterruptedIOException . . . . .	2090
6.416.1.4 InterruptedIOException . . . . .	2090
6.416.1.5 InterruptedIOException . . . . .	2090
6.416.1.6 InterruptedIOException . . . . .	2091
6.416.1.7 ~InterruptedIOException . . . . .	2091
6.416.2 Member Function Documentation . . . . .	2091
6.416.2.1 clone . . . . .	2091
6.417cms::InvalidClientIdException Class Reference . . . . .	2091
6.417.1 Detailed Description . . . . .	2092
6.417.2 Constructor & Destructor Documentation . . . . .	2092
6.417.2.1 InvalidClientIdException . . . . .	2092
6.417.2.2 InvalidClientIdException . . . . .	2092
6.417.2.3 InvalidClientIdException . . . . .	2092
6.417.2.4 InvalidClientIdException . . . . .	2092
6.417.2.5 ~InvalidClientIdException . . . . .	2092
6.418cms::InvalidDestinationException Class Reference . . . . .	2093
6.418.1 Detailed Description . . . . .	2093
6.418.2 Constructor & Destructor Documentation . . . . .	2093
6.418.2.1 InvalidDestinationException . . . . .	2093
6.418.2.2 InvalidDestinationException . . . . .	2093
6.418.2.3 InvalidDestinationException . . . . .	2093
6.418.2.4 InvalidDestinationException . . . . .	2093
6.418.2.5 ~InvalidDestinationException . . . . .	2094
6.419decaf::security::InvalidKeyException Class Reference . . . . .	2094
6.419.1 Constructor & Destructor Documentation . . . . .	2094
6.419.1.1 InvalidKeyException . . . . .	2094
6.419.1.2 InvalidKeyException . . . . .	2095
6.419.1.3 InvalidKeyException . . . . .	2095
6.419.1.4 InvalidKeyException . . . . .	2095

6.419.1.5 InvalidKeyException . . . . .	2095
6.419.1.6 InvalidKeyException . . . . .	2096
6.419.1.7 ~InvalidKeyException . . . . .	2096
6.419.2 Member Function Documentation . . . . .	2096
6.419.2.1 clone . . . . .	2096
6.420decaf::nio::InvalidMarkException Class Reference . . . . .	2096
6.420.1 Constructor & Destructor Documentation . . . . .	2097
6.420.1.1 InvalidMarkException . . . . .	2097
6.420.1.2 InvalidMarkException . . . . .	2097
6.420.1.3 InvalidMarkException . . . . .	2097
6.420.1.4 InvalidMarkException . . . . .	2098
6.420.1.5 InvalidMarkException . . . . .	2098
6.420.1.6 InvalidMarkException . . . . .	2098
6.420.1.7 ~InvalidMarkException . . . . .	2098
6.420.2 Member Function Documentation . . . . .	2098
6.420.2.1 clone . . . . .	2099
6.421cms::InvalidSelectorException Class Reference . . . . .	2099
6.421.1 Detailed Description . . . . .	2099
6.421.2 Constructor & Destructor Documentation . . . . .	2100
6.421.2.1 InvalidSelectorException . . . . .	2100
6.421.2.2 InvalidSelectorException . . . . .	2100
6.421.2.3 InvalidSelectorException . . . . .	2100
6.421.2.4 InvalidSelectorException . . . . .	2100
6.421.2.5 ~InvalidSelectorException . . . . .	2100
6.422decaf::lang::exceptions::InvalidStateException Class Reference . . . . .	2100
6.422.1 Constructor & Destructor Documentation . . . . .	2101
6.422.1.1 InvalidStateException . . . . .	2101
6.422.1.2 InvalidStateException . . . . .	2101
6.422.1.3 InvalidStateException . . . . .	2101
6.422.1.4 InvalidStateException . . . . .	2101
6.422.1.5 InvalidStateException . . . . .	2102
6.422.1.6 InvalidStateException . . . . .	2102
6.422.1.7 ~InvalidStateException . . . . .	2102
6.422.2 Member Function Documentation . . . . .	2102

6.422.2.1 clone . . . . .	2102
6.423decaf::io::IOException Class Reference . . . . .	2103
6.423.1 Constructor & Destructor Documentation . . . . .	2103
6.423.1.1 IOException . . . . .	2103
6.423.1.2 IOException . . . . .	2103
6.423.1.3 IOException . . . . .	2104
6.423.1.4 IOException . . . . .	2104
6.423.1.5 IOException . . . . .	2104
6.423.1.6 IOException . . . . .	2104
6.423.1.7 ~IOException . . . . .	2105
6.423.2 Member Function Documentation . . . . .	2105
6.423.2.1 clone . . . . .	2105
6.424activemq::transport::IOTransport Class Reference . . . . .	2105
6.424.1 Detailed Description . . . . .	2107
6.424.2 Constructor & Destructor Documentation . . . . .	2107
6.424.2.1 IOTransport . . . . .	2107
6.424.2.2 IOTransport . . . . .	2107
6.424.2.3 ~IOTransport . . . . .	2107
6.424.3 Member Function Documentation . . . . .	2107
6.424.3.1 close . . . . .	2107
6.424.3.2 getRemoteAddress . . . . .	2108
6.424.3.3 getTransportListener . . . . .	2108
6.424.3.4 isClosed . . . . .	2108
6.424.3.5 isConnected . . . . .	2108
6.424.3.6 isFaultTolerant . . . . .	2109
6.424.3.7 narrow . . . . .	2109
6.424.3.8 oneway . . . . .	2109
6.424.3.9 reconnect . . . . .	2110
6.424.3.10request . . . . .	2110
6.424.3.11request . . . . .	2110
6.424.3.12run . . . . .	2111
6.424.3.13setInputStream . . . . .	2111
6.424.3.14setOutputStream . . . . .	2111
6.424.3.15setTransportListener . . . . .	2111

6.424.3.16	setWireFormat . . . . .	2112
6.424.3.17	start . . . . .	2112
6.424.3.18	stop . . . . .	2112
6.425	decaf::lang::Iterable< E > Class Template Reference . . . . .	2112
6.425.1	Detailed Description . . . . .	2113
6.425.2	Constructor & Destructor Documentation . . . . .	2113
6.425.2.1	~Iterable . . . . .	2113
6.425.3	Member Function Documentation . . . . .	2113
6.425.3.1	iterator . . . . .	2113
6.425.3.2	iterator . . . . .	2114
6.426	decaf::util::Iterator< T > Class Template Reference . . . . .	2114
6.426.1	Detailed Description . . . . .	2115
6.426.2	Constructor & Destructor Documentation . . . . .	2115
6.426.2.1	~Iterator . . . . .	2115
6.426.3	Member Function Documentation . . . . .	2115
6.426.3.1	hasNext . . . . .	2115
6.426.3.2	next . . . . .	2115
6.426.3.3	remove . . . . .	2115
6.427	activemq::commands::JournalQueueAck Class Reference . . . . .	2116
6.427.1	Constructor & Destructor Documentation . . . . .	2117
6.427.1.1	JournalQueueAck . . . . .	2117
6.427.1.2	~JournalQueueAck . . . . .	2117
6.427.2	Member Function Documentation . . . . .	2117
6.427.2.1	cloneDataStructure . . . . .	2117
6.427.2.2	copyDataStructure . . . . .	2117
6.427.2.3	equals . . . . .	2118
6.427.2.4	getDataStructureType . . . . .	2118
6.427.2.5	getDestination . . . . .	2118
6.427.2.6	getDestination . . . . .	2118
6.427.2.7	getMessageAck . . . . .	2118
6.427.2.8	getMessageAck . . . . .	2118
6.427.2.9	setDestination . . . . .	2118
6.427.2.10	setMessageAck . . . . .	2118
6.427.2.11	toString . . . . .	2119

6.427.3 Field Documentation . . . . .	2119
6.427.3.1 destination . . . . .	2119
6.427.3.2 ID_JOURNALQUEUEACK . . . . .	2119
6.427.3.3 messageAck . . . . .	2119
6.428activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller	
Class Reference . . . . .	2119
6.428.1 Detailed Description . . . . .	2120
6.428.2 Constructor & Destructor Documentation . . . . .	2120
6.428.2.1 JournalQueueAckMarshaller . . . . .	2120
6.428.2.2 ~JournalQueueAckMarshaller . . . . .	2120
6.428.3 Member Function Documentation . . . . .	2120
6.428.3.1 createObject . . . . .	2120
6.428.3.2 getDataStructureType . . . . .	2121
6.428.3.3 looseMarshal . . . . .	2121
6.428.3.4 looseUnmarshal . . . . .	2121
6.428.3.5 tightMarshal1 . . . . .	2122
6.428.3.6 tightMarshal2 . . . . .	2122
6.428.3.7 tightUnmarshal . . . . .	2123
6.429activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	
Class Reference . . . . .	2123
6.429.1 Detailed Description . . . . .	2124
6.429.2 Constructor & Destructor Documentation . . . . .	2124
6.429.2.1 JournalQueueAckMarshaller . . . . .	2124
6.429.2.2 ~JournalQueueAckMarshaller . . . . .	2124
6.429.3 Member Function Documentation . . . . .	2124
6.429.3.1 createObject . . . . .	2124
6.429.3.2 getDataStructureType . . . . .	2125
6.429.3.3 looseMarshal . . . . .	2125
6.429.3.4 looseUnmarshal . . . . .	2125
6.429.3.5 tightMarshal1 . . . . .	2126
6.429.3.6 tightMarshal2 . . . . .	2126
6.429.3.7 tightUnmarshal . . . . .	2127
6.430activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	
Class Reference . . . . .	2127
6.430.1 Detailed Description . . . . .	2128

6.430.2 Constructor & Destructor Documentation . . . . .	2128
6.430.2.1 JournalQueueAckMarshaller . . . . .	2128
6.430.2.2 ~JournalQueueAckMarshaller . . . . .	2128
6.430.3 Member Function Documentation . . . . .	2128
6.430.3.1 createObject . . . . .	2128
6.430.3.2 getDataStructureType . . . . .	2129
6.430.3.3 looseMarshal . . . . .	2129
6.430.3.4 looseUnmarshal . . . . .	2129
6.430.3.5 tightMarshal1 . . . . .	2130
6.430.3.6 tightMarshal2 . . . . .	2130
6.430.3.7 tightUnmarshal . . . . .	2131
6.431activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	
Class Reference . . . . .	2131
6.431.1 Detailed Description . . . . .	2132
6.431.2 Constructor & Destructor Documentation . . . . .	2132
6.431.2.1 JournalQueueAckMarshaller . . . . .	2132
6.431.2.2 ~JournalQueueAckMarshaller . . . . .	2132
6.431.3 Member Function Documentation . . . . .	2132
6.431.3.1 createObject . . . . .	2132
6.431.3.2 getDataStructureType . . . . .	2133
6.431.3.3 looseMarshal . . . . .	2133
6.431.3.4 looseUnmarshal . . . . .	2133
6.431.3.5 tightMarshal1 . . . . .	2134
6.431.3.6 tightMarshal2 . . . . .	2134
6.431.3.7 tightUnmarshal . . . . .	2135
6.432activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	
Class Reference . . . . .	2135
6.432.1 Detailed Description . . . . .	2136
6.432.2 Constructor & Destructor Documentation . . . . .	2136
6.432.2.1 JournalQueueAckMarshaller . . . . .	2136
6.432.2.2 ~JournalQueueAckMarshaller . . . . .	2136
6.432.3 Member Function Documentation . . . . .	2136
6.432.3.1 createObject . . . . .	2136
6.432.3.2 getDataStructureType . . . . .	2137



6.432.3.3 looseMarshal . . . . .	2137
6.432.3.4 looseUnmarshal . . . . .	2137
6.432.3.5 tightMarshal1 . . . . .	2138
6.432.3.6 tightMarshal2 . . . . .	2138
6.432.3.7 tightUnmarshal . . . . .	2139
6.433activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	
Class Reference . . . . .	2139
6.433.1 Detailed Description . . . . .	2140
6.433.2 Constructor & Destructor Documentation . . . . .	2140
6.433.2.1 JournalQueueAckMarshaller . . . . .	2140
6.433.2.2 ~JournalQueueAckMarshaller . . . . .	2140
6.433.3 Member Function Documentation . . . . .	2140
6.433.3.1 createObject . . . . .	2140
6.433.3.2 getDataStructureType . . . . .	2141
6.433.3.3 looseMarshal . . . . .	2141
6.433.3.4 looseUnmarshal . . . . .	2141
6.433.3.5 tightMarshal1 . . . . .	2142
6.433.3.6 tightMarshal2 . . . . .	2142
6.433.3.7 tightUnmarshal . . . . .	2143
6.434activemq::commands::JournalTopicAck Class Reference . . . . .	2143
6.434.1 Constructor & Destructor Documentation . . . . .	2144
6.434.1.1 JournalTopicAck . . . . .	2144
6.434.1.2 ~JournalTopicAck . . . . .	2144
6.434.2 Member Function Documentation . . . . .	2144
6.434.2.1 cloneDataStructure . . . . .	2145
6.434.2.2 copyDataStructure . . . . .	2145
6.434.2.3 equals . . . . .	2145
6.434.2.4 getClientId . . . . .	2145
6.434.2.5 getClientId . . . . .	2145
6.434.2.6 getDataStructureType . . . . .	2145
6.434.2.7 getDestination . . . . .	2146
6.434.2.8 getDestination . . . . .	2146
6.434.2.9 getMessageld . . . . .	2146
6.434.2.10getMessageld . . . . .	2146

6.434.2.11	getMessageSequenceId . . . . .	2146
6.434.2.12	getSubscriptionName . . . . .	2146
6.434.2.13	getSubscriptionName . . . . .	2146
6.434.2.14	getTransactionId . . . . .	2146
6.434.2.15	getTransactionId . . . . .	2146
6.434.2.16	setClientId . . . . .	2146
6.434.2.17	setDestination . . . . .	2146
6.434.2.18	setMessageId . . . . .	2146
6.434.2.19	setMessageSequenceId . . . . .	2146
6.434.2.20	setSubscriptionName . . . . .	2147
6.434.2.21	setTransactionId . . . . .	2147
6.434.2.22	toString . . . . .	2147
6.434.3	Field Documentation . . . . .	2147
6.434.3.1	clientId . . . . .	2147
6.434.3.2	destination . . . . .	2147
6.434.3.3	ID_JOURNALTOPICACK . . . . .	2147
6.434.3.4	messageId . . . . .	2147
6.434.3.5	messageSequenceId . . . . .	2147
6.434.3.6	subscriptionName . . . . .	2147
6.434.3.7	transactionId . . . . .	2147
6.435	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	
	Class Reference . . . . .	2148
6.435.1	Detailed Description . . . . .	2148
6.435.2	Constructor & Destructor Documentation . . . . .	2149
6.435.2.1	JournalTopicAckMarshaller . . . . .	2149
6.435.2.2	~JournalTopicAckMarshaller . . . . .	2149
6.435.3	Member Function Documentation . . . . .	2149
6.435.3.1	createObject . . . . .	2149
6.435.3.2	getDataStructureType . . . . .	2149
6.435.3.3	looseMarshal . . . . .	2149
6.435.3.4	looseUnmarshal . . . . .	2150
6.435.3.5	tightMarshal1 . . . . .	2150
6.435.3.6	tightMarshal2 . . . . .	2151
6.435.3.7	tightUnmarshal . . . . .	2151

6.436activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	
Class Reference . . . . .	2152
6.436.1 Detailed Description . . . . .	2152
6.436.2 Constructor & Destructor Documentation . . . . .	2153
6.436.2.1 JournalTopicAckMarshaller . . . . .	2153
6.436.2.2 ~JournalTopicAckMarshaller . . . . .	2153
6.436.3 Member Function Documentation . . . . .	2153
6.436.3.1 createObject . . . . .	2153
6.436.3.2 getDataStructureType . . . . .	2153
6.436.3.3 looseMarshal . . . . .	2153
6.436.3.4 looseUnmarshal . . . . .	2154
6.436.3.5 tightMarshal1 . . . . .	2154
6.436.3.6 tightMarshal2 . . . . .	2155
6.436.3.7 tightUnmarshal . . . . .	2155
6.437activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	
Class Reference . . . . .	2156
6.437.1 Detailed Description . . . . .	2156
6.437.2 Constructor & Destructor Documentation . . . . .	2157
6.437.2.1 JournalTopicAckMarshaller . . . . .	2157
6.437.2.2 ~JournalTopicAckMarshaller . . . . .	2157
6.437.3 Member Function Documentation . . . . .	2157
6.437.3.1 createObject . . . . .	2157
6.437.3.2 getDataStructureType . . . . .	2157
6.437.3.3 looseMarshal . . . . .	2157
6.437.3.4 looseUnmarshal . . . . .	2158
6.437.3.5 tightMarshal1 . . . . .	2158
6.437.3.6 tightMarshal2 . . . . .	2159
6.437.3.7 tightUnmarshal . . . . .	2159
6.438activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller	
Class Reference . . . . .	2160
6.438.1 Detailed Description . . . . .	2160
6.438.2 Constructor & Destructor Documentation . . . . .	2161
6.438.2.1 JournalTopicAckMarshaller . . . . .	2161
6.438.2.2 ~JournalTopicAckMarshaller . . . . .	2161
6.438.3 Member Function Documentation . . . . .	2161

6.438.3.1	createObject . . . . .	2161
6.438.3.2	getDataStructureType . . . . .	2161
6.438.3.3	looseMarshal . . . . .	2161
6.438.3.4	looseUnmarshal . . . . .	2162
6.438.3.5	tightMarshal1 . . . . .	2162
6.438.3.6	tightMarshal2 . . . . .	2163
6.438.3.7	tightUnmarshal . . . . .	2163
6.439	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	
	Class Reference . . . . .	2164
6.439.1	Detailed Description . . . . .	2164
6.439.2	Constructor & Destructor Documentation . . . . .	2165
6.439.2.1	JournalTopicAckMarshaller . . . . .	2165
6.439.2.2	~JournalTopicAckMarshaller . . . . .	2165
6.439.3	Member Function Documentation . . . . .	2165
6.439.3.1	createObject . . . . .	2165
6.439.3.2	getDataStructureType . . . . .	2165
6.439.3.3	looseMarshal . . . . .	2165
6.439.3.4	looseUnmarshal . . . . .	2166
6.439.3.5	tightMarshal1 . . . . .	2166
6.439.3.6	tightMarshal2 . . . . .	2167
6.439.3.7	tightUnmarshal . . . . .	2167
6.440	activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	
	Class Reference . . . . .	2168
6.440.1	Detailed Description . . . . .	2168
6.440.2	Constructor & Destructor Documentation . . . . .	2169
6.440.2.1	JournalTopicAckMarshaller . . . . .	2169
6.440.2.2	~JournalTopicAckMarshaller . . . . .	2169
6.440.3	Member Function Documentation . . . . .	2169
6.440.3.1	createObject . . . . .	2169
6.440.3.2	getDataStructureType . . . . .	2169
6.440.3.3	looseMarshal . . . . .	2169
6.440.3.4	looseUnmarshal . . . . .	2170
6.440.3.5	tightMarshal1 . . . . .	2170
6.440.3.6	tightMarshal2 . . . . .	2171

6.440.3.7	tightUnmarshal . . . . .	2171
6.441	activemq::commands::JournalTrace Class Reference . . . . .	2171
6.441.1	Constructor & Destructor Documentation . . . . .	2172
6.441.1.1	JournalTrace . . . . .	2172
6.441.1.2	~JournalTrace . . . . .	2172
6.441.2	Member Function Documentation . . . . .	2172
6.441.2.1	cloneDataStructure . . . . .	2173
6.441.2.2	copyDataStructure . . . . .	2173
6.441.2.3	equals . . . . .	2173
6.441.2.4	getDataStructureType . . . . .	2173
6.441.2.5	getMessage . . . . .	2174
6.441.2.6	getMessage . . . . .	2174
6.441.2.7	setMessage . . . . .	2174
6.441.2.8	toString . . . . .	2174
6.441.3	Field Documentation . . . . .	2174
6.441.3.1	ID_JOURNALTRACE . . . . .	2174
6.441.3.2	message . . . . .	2174
6.442	activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller Class Reference . . . . .	2174
6.442.1	Detailed Description . . . . .	2175
6.442.2	Constructor & Destructor Documentation . . . . .	2175
6.442.2.1	JournalTraceMarshaller . . . . .	2175
6.442.2.2	~JournalTraceMarshaller . . . . .	2175
6.442.3	Member Function Documentation . . . . .	2175
6.442.3.1	createObject . . . . .	2175
6.442.3.2	getDataStructureType . . . . .	2176
6.442.3.3	looseMarshal . . . . .	2176
6.442.3.4	looseUnmarshal . . . . .	2176
6.442.3.5	tightMarshal1 . . . . .	2177
6.442.3.6	tightMarshal2 . . . . .	2177
6.442.3.7	tightUnmarshal . . . . .	2178
6.443	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller Class Reference . . . . .	2178
6.443.1	Detailed Description . . . . .	2179

6.443.2 Constructor & Destructor Documentation . . . . .	2179
6.443.2.1 JournalTraceMarshaller . . . . .	2179
6.443.2.2 ~JournalTraceMarshaller . . . . .	2179
6.443.3 Member Function Documentation . . . . .	2179
6.443.3.1 createObject . . . . .	2179
6.443.3.2 getDataStructureType . . . . .	2180
6.443.3.3 looseMarshal . . . . .	2180
6.443.3.4 looseUnmarshal . . . . .	2180
6.443.3.5 tightMarshal1 . . . . .	2181
6.443.3.6 tightMarshal2 . . . . .	2181
6.443.3.7 tightUnmarshal . . . . .	2182
6.444activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller Class	
Reference . . . . .	2182
6.444.1 Detailed Description . . . . .	2183
6.444.2 Constructor & Destructor Documentation . . . . .	2183
6.444.2.1 JournalTraceMarshaller . . . . .	2183
6.444.2.2 ~JournalTraceMarshaller . . . . .	2183
6.444.3 Member Function Documentation . . . . .	2183
6.444.3.1 createObject . . . . .	2183
6.444.3.2 getDataStructureType . . . . .	2184
6.444.3.3 looseMarshal . . . . .	2184
6.444.3.4 looseUnmarshal . . . . .	2184
6.444.3.5 tightMarshal1 . . . . .	2185
6.444.3.6 tightMarshal2 . . . . .	2185
6.444.3.7 tightUnmarshal . . . . .	2186
6.445activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class	
Reference . . . . .	2186
6.445.1 Detailed Description . . . . .	2187
6.445.2 Constructor & Destructor Documentation . . . . .	2187
6.445.2.1 JournalTraceMarshaller . . . . .	2187
6.445.2.2 ~JournalTraceMarshaller . . . . .	2187
6.445.3 Member Function Documentation . . . . .	2187
6.445.3.1 createObject . . . . .	2187
6.445.3.2 getDataStructureType . . . . .	2188

6.445.3.3 looseMarshal . . . . .	2188
6.445.3.4 looseUnmarshal . . . . .	2188
6.445.3.5 tightMarshal1 . . . . .	2189
6.445.3.6 tightMarshal2 . . . . .	2189
6.445.3.7 tightUnmarshal . . . . .	2190
6.446activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference . . . . .	2190
6.446.1 Detailed Description . . . . .	2191
6.446.2 Constructor & Destructor Documentation . . . . .	2191
6.446.2.1 JournalTraceMarshaller . . . . .	2191
6.446.2.2 ~JournalTraceMarshaller . . . . .	2191
6.446.3 Member Function Documentation . . . . .	2191
6.446.3.1 createObject . . . . .	2191
6.446.3.2 getDataStructureType . . . . .	2192
6.446.3.3 looseMarshal . . . . .	2192
6.446.3.4 looseUnmarshal . . . . .	2192
6.446.3.5 tightMarshal1 . . . . .	2193
6.446.3.6 tightMarshal2 . . . . .	2193
6.446.3.7 tightUnmarshal . . . . .	2194
6.447activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference . . . . .	2194
6.447.1 Detailed Description . . . . .	2195
6.447.2 Constructor & Destructor Documentation . . . . .	2195
6.447.2.1 JournalTraceMarshaller . . . . .	2195
6.447.2.2 ~JournalTraceMarshaller . . . . .	2195
6.447.3 Member Function Documentation . . . . .	2195
6.447.3.1 createObject . . . . .	2195
6.447.3.2 getDataStructureType . . . . .	2196
6.447.3.3 looseMarshal . . . . .	2196
6.447.3.4 looseUnmarshal . . . . .	2196
6.447.3.5 tightMarshal1 . . . . .	2197
6.447.3.6 tightMarshal2 . . . . .	2197
6.447.3.7 tightUnmarshal . . . . .	2198
6.448activemq::commands::JournalTransaction Class Reference . . . . .	2198

6.448.1 Constructor & Destructor Documentation . . . . .	2199
6.448.1.1 JournalTransaction . . . . .	2199
6.448.1.2 ~JournalTransaction . . . . .	2199
6.448.2 Member Function Documentation . . . . .	2199
6.448.2.1 cloneDataStructure . . . . .	2199
6.448.2.2 copyDataStructure . . . . .	2200
6.448.2.3 equals . . . . .	2200
6.448.2.4 getDataStructureType . . . . .	2200
6.448.2.5 getTransactionId . . . . .	2200
6.448.2.6 getTransactionId . . . . .	2200
6.448.2.7 getType . . . . .	2200
6.448.2.8 getWasPrepared . . . . .	2201
6.448.2.9 setTransactionId . . . . .	2201
6.448.2.10 setType . . . . .	2201
6.448.2.11 setWasPrepared . . . . .	2201
6.448.2.12 toString . . . . .	2201
6.448.3 Field Documentation . . . . .	2201
6.448.3.1 ID_JOURNALTRANSACTION . . . . .	2201
6.448.3.2 transactionId . . . . .	2201
6.448.3.3 type . . . . .	2201
6.448.3.4 wasPrepared . . . . .	2201
6.449activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller	
Class Reference . . . . .	2201
6.449.1 Detailed Description . . . . .	2202
6.449.2 Constructor & Destructor Documentation . . . . .	2202
6.449.2.1 JournalTransactionMarshaller . . . . .	2202
6.449.2.2 ~JournalTransactionMarshaller . . . . .	2203
6.449.3 Member Function Documentation . . . . .	2203
6.449.3.1 createObject . . . . .	2203
6.449.3.2 getDataStructureType . . . . .	2203
6.449.3.3 looseMarshal . . . . .	2203
6.449.3.4 looseUnmarshal . . . . .	2204
6.449.3.5 tightMarshal1 . . . . .	2204
6.449.3.6 tightMarshal2 . . . . .	2204



6.449.3.7	tightUnmarshal . . . . .	2205
6.450	activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	
	Class Reference . . . . .	2205
6.450.1	Detailed Description . . . . .	2206
6.450.2	Constructor & Destructor Documentation . . . . .	2206
6.450.2.1	JournalTransactionMarshaller . . . . .	2206
6.450.2.2	~JournalTransactionMarshaller . . . . .	2206
6.450.3	Member Function Documentation . . . . .	2206
6.450.3.1	createObject . . . . .	2207
6.450.3.2	getDataStructureType . . . . .	2207
6.450.3.3	looseMarshal . . . . .	2207
6.450.3.4	looseUnmarshal . . . . .	2208
6.450.3.5	tightMarshal1 . . . . .	2208
6.450.3.6	tightMarshal2 . . . . .	2208
6.450.3.7	tightUnmarshal . . . . .	2209
6.451	activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	
	Class Reference . . . . .	2209
6.451.1	Detailed Description . . . . .	2210
6.451.2	Constructor & Destructor Documentation . . . . .	2210
6.451.2.1	JournalTransactionMarshaller . . . . .	2210
6.451.2.2	~JournalTransactionMarshaller . . . . .	2210
6.451.3	Member Function Documentation . . . . .	2210
6.451.3.1	createObject . . . . .	2211
6.451.3.2	getDataStructureType . . . . .	2211
6.451.3.3	looseMarshal . . . . .	2211
6.451.3.4	looseUnmarshal . . . . .	2212
6.451.3.5	tightMarshal1 . . . . .	2212
6.451.3.6	tightMarshal2 . . . . .	2212
6.451.3.7	tightUnmarshal . . . . .	2213
6.452	activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	
	Class Reference . . . . .	2213
6.452.1	Detailed Description . . . . .	2214
6.452.2	Constructor & Destructor Documentation . . . . .	2214
6.452.2.1	JournalTransactionMarshaller . . . . .	2214
6.452.2.2	~JournalTransactionMarshaller . . . . .	2214

6.452.3 Member Function Documentation . . . . .	2214
6.452.3.1 createObject . . . . .	2215
6.452.3.2 getDataStructureType . . . . .	2215
6.452.3.3 looseMarshal . . . . .	2215
6.452.3.4 looseUnmarshal . . . . .	2216
6.452.3.5 tightMarshal1 . . . . .	2216
6.452.3.6 tightMarshal2 . . . . .	2216
6.452.3.7 tightUnmarshal . . . . .	2217
6.453activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	
Class Reference . . . . .	2217
6.453.1 Detailed Description . . . . .	2218
6.453.2 Constructor & Destructor Documentation . . . . .	2218
6.453.2.1 JournalTransactionMarshaller . . . . .	2218
6.453.2.2 ~JournalTransactionMarshaller . . . . .	2218
6.453.3 Member Function Documentation . . . . .	2218
6.453.3.1 createObject . . . . .	2219
6.453.3.2 getDataStructureType . . . . .	2219
6.453.3.3 looseMarshal . . . . .	2219
6.453.3.4 looseUnmarshal . . . . .	2220
6.453.3.5 tightMarshal1 . . . . .	2220
6.453.3.6 tightMarshal2 . . . . .	2220
6.453.3.7 tightUnmarshal . . . . .	2221
6.454activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	
Class Reference . . . . .	2221
6.454.1 Detailed Description . . . . .	2222
6.454.2 Constructor & Destructor Documentation . . . . .	2222
6.454.2.1 JournalTransactionMarshaller . . . . .	2222
6.454.2.2 ~JournalTransactionMarshaller . . . . .	2222
6.454.3 Member Function Documentation . . . . .	2222
6.454.3.1 createObject . . . . .	2223
6.454.3.2 getDataStructureType . . . . .	2223
6.454.3.3 looseMarshal . . . . .	2223
6.454.3.4 looseUnmarshal . . . . .	2224
6.454.3.5 tightMarshal1 . . . . .	2224

6.454.3.6	tightMarshal2 . . . . .	2224
6.454.3.7	tightUnmarshal . . . . .	2225
6.455	activemq::commands::KeepAliveInfo Class Reference . . . . .	2225
6.455.1	Constructor & Destructor Documentation . . . . .	2226
6.455.1.1	KeepAliveInfo . . . . .	2226
6.455.1.2	~KeepAliveInfo . . . . .	2226
6.455.2	Member Function Documentation . . . . .	2226
6.455.2.1	cloneDataStructure . . . . .	2226
6.455.2.2	copyDataStructure . . . . .	2227
6.455.2.3	equals . . . . .	2227
6.455.2.4	getDataStructureType . . . . .	2227
6.455.2.5	isKeepAliveInfo . . . . .	2227
6.455.2.6	toString . . . . .	2228
6.455.2.7	visit . . . . .	2228
6.455.3	Field Documentation . . . . .	2228
6.455.3.1	ID_KEEPLIVEINFO . . . . .	2228
6.456	activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller Class Reference . . . . .	2228
6.456.1	Detailed Description . . . . .	2229
6.456.2	Constructor & Destructor Documentation . . . . .	2229
6.456.2.1	KeepAliveInfoMarshaller . . . . .	2229
6.456.2.2	~KeepAliveInfoMarshaller . . . . .	2229
6.456.3	Member Function Documentation . . . . .	2229
6.456.3.1	createObject . . . . .	2230
6.456.3.2	getDataStructureType . . . . .	2230
6.456.3.3	looseMarshal . . . . .	2230
6.456.3.4	looseUnmarshal . . . . .	2231
6.456.3.5	tightMarshal1 . . . . .	2231
6.456.3.6	tightMarshal2 . . . . .	2232
6.456.3.7	tightUnmarshal . . . . .	2232
6.457	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference . . . . .	2233
6.457.1	Detailed Description . . . . .	2233
6.457.2	Constructor & Destructor Documentation . . . . .	2234

6.457.2.1 KeepAliveInfoMarshaller . . . . .	2234
6.457.2.2 ~KeepAliveInfoMarshaller . . . . .	2234
6.457.3 Member Function Documentation . . . . .	2234
6.457.3.1 createObject . . . . .	2234
6.457.3.2 getDataStructureType . . . . .	2234
6.457.3.3 looseMarshal . . . . .	2234
6.457.3.4 looseUnmarshal . . . . .	2235
6.457.3.5 tightMarshal1 . . . . .	2235
6.457.3.6 tightMarshal2 . . . . .	2236
6.457.3.7 tightUnmarshal . . . . .	2236
6.458activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class	
Reference . . . . .	2237
6.458.1 Detailed Description . . . . .	2237
6.458.2 Constructor & Destructor Documentation . . . . .	2238
6.458.2.1 KeepAliveInfoMarshaller . . . . .	2238
6.458.2.2 ~KeepAliveInfoMarshaller . . . . .	2238
6.458.3 Member Function Documentation . . . . .	2238
6.458.3.1 createObject . . . . .	2238
6.458.3.2 getDataStructureType . . . . .	2238
6.458.3.3 looseMarshal . . . . .	2238
6.458.3.4 looseUnmarshal . . . . .	2239
6.458.3.5 tightMarshal1 . . . . .	2239
6.458.3.6 tightMarshal2 . . . . .	2240
6.458.3.7 tightUnmarshal . . . . .	2240
6.459activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class	
Reference . . . . .	2241
6.459.1 Detailed Description . . . . .	2241
6.459.2 Constructor & Destructor Documentation . . . . .	2242
6.459.2.1 KeepAliveInfoMarshaller . . . . .	2242
6.459.2.2 ~KeepAliveInfoMarshaller . . . . .	2242
6.459.3 Member Function Documentation . . . . .	2242
6.459.3.1 createObject . . . . .	2242
6.459.3.2 getDataStructureType . . . . .	2242
6.459.3.3 looseMarshal . . . . .	2242

6.459.3.4 looseUnmarshal . . . . .	2243
6.459.3.5 tightMarshal1 . . . . .	2243
6.459.3.6 tightMarshal2 . . . . .	2244
6.459.3.7 tightUnmarshal . . . . .	2244
6.460activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class	
Reference . . . . .	2245
6.460.1 Detailed Description . . . . .	2245
6.460.2 Constructor & Destructor Documentation . . . . .	2246
6.460.2.1 KeepAliveInfoMarshaller . . . . .	2246
6.460.2.2 ~KeepAliveInfoMarshaller . . . . .	2246
6.460.3 Member Function Documentation . . . . .	2246
6.460.3.1 createObject . . . . .	2246
6.460.3.2 getDataStructureType . . . . .	2246
6.460.3.3 looseMarshal . . . . .	2246
6.460.3.4 looseUnmarshal . . . . .	2247
6.460.3.5 tightMarshal1 . . . . .	2247
6.460.3.6 tightMarshal2 . . . . .	2248
6.460.3.7 tightUnmarshal . . . . .	2248
6.461activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class	
Reference . . . . .	2249
6.461.1 Detailed Description . . . . .	2249
6.461.2 Constructor & Destructor Documentation . . . . .	2250
6.461.2.1 KeepAliveInfoMarshaller . . . . .	2250
6.461.2.2 ~KeepAliveInfoMarshaller . . . . .	2250
6.461.3 Member Function Documentation . . . . .	2250
6.461.3.1 createObject . . . . .	2250
6.461.3.2 getDataStructureType . . . . .	2250
6.461.3.3 looseMarshal . . . . .	2250
6.461.3.4 looseUnmarshal . . . . .	2251
6.461.3.5 tightMarshal1 . . . . .	2251
6.461.3.6 tightMarshal2 . . . . .	2252
6.461.3.7 tightUnmarshal . . . . .	2252
6.462decaf::security::Key Class Reference . . . . .	2253
6.462.1 Detailed Description . . . . .	2253

6.462.2 Constructor & Destructor Documentation . . . . .	2254
6.462.2.1 ~Key . . . . .	2254
6.462.3 Member Function Documentation . . . . .	2254
6.462.3.1 getAlgorithm . . . . .	2254
6.462.3.2 getEncoded . . . . .	2254
6.462.3.3 getFormat . . . . .	2254
6.463decaf::security::KeyException Class Reference . . . . .	2255
6.463.1 Constructor & Destructor Documentation . . . . .	2255
6.463.1.1 KeyException . . . . .	2255
6.463.1.2 KeyException . . . . .	2256
6.463.1.3 KeyException . . . . .	2256
6.463.1.4 KeyException . . . . .	2256
6.463.1.5 KeyException . . . . .	2256
6.463.1.6 KeyException . . . . .	2257
6.463.1.7 ~KeyException . . . . .	2257
6.463.2 Member Function Documentation . . . . .	2257
6.463.2.1 clone . . . . .	2257
6.464decaf::security::KeyManagementException Class Reference . . . . .	2257
6.464.1 Constructor & Destructor Documentation . . . . .	2258
6.464.1.1 KeyManagementException . . . . .	2258
6.464.1.2 KeyManagementException . . . . .	2258
6.464.1.3 KeyManagementException . . . . .	2258
6.464.1.4 KeyManagementException . . . . .	2259
6.464.1.5 KeyManagementException . . . . .	2259
6.464.1.6 KeyManagementException . . . . .	2259
6.464.1.7 ~KeyManagementException . . . . .	2259
6.464.2 Member Function Documentation . . . . .	2259
6.464.2.1 clone . . . . .	2260
6.465activemq::commands::LastPartialCommand Class Reference . . . . .	2260
6.465.1 Constructor & Destructor Documentation . . . . .	2261
6.465.1.1 LastPartialCommand . . . . .	2261
6.465.1.2 ~LastPartialCommand . . . . .	2261
6.465.2 Member Function Documentation . . . . .	2261
6.465.2.1 cloneDataStructure . . . . .	2261

6.465.2.2	copyDataStructure . . . . .	2261
6.465.2.3	equals . . . . .	2261
6.465.2.4	getDataStructureType . . . . .	2262
6.465.2.5	toString . . . . .	2262
6.465.3	Field Documentation . . . . .	2262
6.465.3.1	ID_LASTPARTIALCOMMAND . . . . .	2262
6.466	activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller	
	Class Reference . . . . .	2262
6.466.1	Detailed Description . . . . .	2263
6.466.2	Constructor & Destructor Documentation . . . . .	2263
6.466.2.1	LastPartialCommandMarshaller . . . . .	2263
6.466.2.2	~LastPartialCommandMarshaller . . . . .	2263
6.466.3	Member Function Documentation . . . . .	2263
6.466.3.1	createObject . . . . .	2264
6.466.3.2	getDataStructureType . . . . .	2264
6.466.3.3	looseMarshal . . . . .	2264
6.466.3.4	looseUnmarshal . . . . .	2265
6.466.3.5	tightMarshal1 . . . . .	2265
6.466.3.6	tightMarshal2 . . . . .	2266
6.466.3.7	tightUnmarshal . . . . .	2266
6.467	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
	Class Reference . . . . .	2267
6.467.1	Detailed Description . . . . .	2267
6.467.2	Constructor & Destructor Documentation . . . . .	2268
6.467.2.1	LastPartialCommandMarshaller . . . . .	2268
6.467.2.2	~LastPartialCommandMarshaller . . . . .	2268
6.467.3	Member Function Documentation . . . . .	2268
6.467.3.1	createObject . . . . .	2268
6.467.3.2	getDataStructureType . . . . .	2268
6.467.3.3	looseMarshal . . . . .	2268
6.467.3.4	looseUnmarshal . . . . .	2269
6.467.3.5	tightMarshal1 . . . . .	2269
6.467.3.6	tightMarshal2 . . . . .	2270
6.467.3.7	tightUnmarshal . . . . .	2270

6.468activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	
Class Reference . . . . .	2271
6.468.1 Detailed Description . . . . .	2271
6.468.2 Constructor & Destructor Documentation . . . . .	2272
6.468.2.1 LastPartialCommandMarshaller . . . . .	2272
6.468.2.2 ~LastPartialCommandMarshaller . . . . .	2272
6.468.3 Member Function Documentation . . . . .	2272
6.468.3.1 createObject . . . . .	2272
6.468.3.2 getDataStructureType . . . . .	2272
6.468.3.3 looseMarshal . . . . .	2272
6.468.3.4 looseUnmarshal . . . . .	2273
6.468.3.5 tightMarshal1 . . . . .	2273
6.468.3.6 tightMarshal2 . . . . .	2274
6.468.3.7 tightUnmarshal . . . . .	2274
6.469activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	
Class Reference . . . . .	2275
6.469.1 Detailed Description . . . . .	2275
6.469.2 Constructor & Destructor Documentation . . . . .	2276
6.469.2.1 LastPartialCommandMarshaller . . . . .	2276
6.469.2.2 ~LastPartialCommandMarshaller . . . . .	2276
6.469.3 Member Function Documentation . . . . .	2276
6.469.3.1 createObject . . . . .	2276
6.469.3.2 getDataStructureType . . . . .	2276
6.469.3.3 looseMarshal . . . . .	2276
6.469.3.4 looseUnmarshal . . . . .	2277
6.469.3.5 tightMarshal1 . . . . .	2277
6.469.3.6 tightMarshal2 . . . . .	2278
6.469.3.7 tightUnmarshal . . . . .	2278
6.470activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
Class Reference . . . . .	2279
6.470.1 Detailed Description . . . . .	2279
6.470.2 Constructor & Destructor Documentation . . . . .	2280
6.470.2.1 LastPartialCommandMarshaller . . . . .	2280
6.470.2.2 ~LastPartialCommandMarshaller . . . . .	2280
6.470.3 Member Function Documentation . . . . .	2280



6.470.3.1	createObject . . . . .	2280
6.470.3.2	getDataStructureType . . . . .	2280
6.470.3.3	looseMarshal . . . . .	2280
6.470.3.4	looseUnmarshal . . . . .	2281
6.470.3.5	tightMarshal1 . . . . .	2281
6.470.3.6	tightMarshal2 . . . . .	2282
6.470.3.7	tightUnmarshal . . . . .	2282
6.471	activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	
	Class Reference . . . . .	2283
6.471.1	Detailed Description . . . . .	2283
6.471.2	Constructor & Destructor Documentation . . . . .	2284
6.471.2.1	LastPartialCommandMarshaller . . . . .	2284
6.471.2.2	~LastPartialCommandMarshaller . . . . .	2284
6.471.3	Member Function Documentation . . . . .	2284
6.471.3.1	createObject . . . . .	2284
6.471.3.2	getDataStructureType . . . . .	2284
6.471.3.3	looseMarshal . . . . .	2284
6.471.3.4	looseUnmarshal . . . . .	2285
6.471.3.5	tightMarshal1 . . . . .	2285
6.471.3.6	tightMarshal2 . . . . .	2286
6.471.3.7	tightUnmarshal . . . . .	2286
6.472	decaf::util::comparators::Less< E > Class Template Reference . . . . .	2287
6.472.1	Detailed Description . . . . .	2287
6.472.2	Constructor & Destructor Documentation . . . . .	2287
6.472.2.1	Less . . . . .	2287
6.472.2.2	~Less . . . . .	2287
6.472.3	Member Function Documentation . . . . .	2287
6.472.3.1	compare . . . . .	2288
6.472.3.2	operator() . . . . .	2288
6.473	std::less< decaf::lang::ArrayPointer< T > > Struct Template Reference . . . . .	2289
6.473.1	Detailed Description . . . . .	2289
6.473.2	Member Function Documentation . . . . .	2289
6.473.2.1	operator() . . . . .	2289
6.474	std::less< decaf::lang::Pointer< T > > Struct Template Reference . . . . .	2289

6.474.1 Detailed Description . . . . .	2290
6.474.2 Member Function Documentation . . . . .	2290
6.474.2.1 operator() . . . . .	2290
6.475decaf::util::logging::Level Class Reference . . . . .	2290
6.475.1 Detailed Description . . . . .	2292
6.475.2 Constructor & Destructor Documentation . . . . .	2292
6.475.2.1 Level . . . . .	2292
6.475.2.2 ~Level . . . . .	2292
6.475.3 Member Function Documentation . . . . .	2292
6.475.3.1 compareTo . . . . .	2292
6.475.3.2 equals . . . . .	2292
6.475.3.3 getName . . . . .	2293
6.475.3.4 intValue . . . . .	2293
6.475.3.5 operator< . . . . .	2293
6.475.3.6 operator== . . . . .	2293
6.475.3.7 parse . . . . .	2293
6.475.3.8 toString . . . . .	2293
6.475.4 Field Documentation . . . . .	2294
6.475.4.1 ALL . . . . .	2294
6.475.4.2 CONFIG . . . . .	2294
6.475.4.3 DEBUG . . . . .	2294
6.475.4.4 FINE . . . . .	2294
6.475.4.5 FINER . . . . .	2294
6.475.4.6 FINEST . . . . .	2295
6.475.4.7 INFO . . . . .	2295
6.475.4.8 INHERIT . . . . .	2295
6.475.4.9 OFF . . . . .	2295
6.475.4.10SEVERE . . . . .	2295
6.475.4.11WARNING . . . . .	2295
6.476decaf::util::List< E > Class Template Reference . . . . .	2296
6.476.1 Detailed Description . . . . .	2297
6.476.2 Constructor & Destructor Documentation . . . . .	2297
6.476.2.1 List . . . . .	2297
6.476.2.2 ~List . . . . .	2297

6.476.3 Member Function Documentation . . . . .	2297
6.476.3.1 add . . . . .	2297
6.476.3.2 addAll . . . . .	2298
6.476.3.3 get . . . . .	2299
6.476.3.4 indexOf . . . . .	2299
6.476.3.5 lastIndexOf . . . . .	2300
6.476.3.6 listIterator . . . . .	2300
6.476.3.7 listIterator . . . . .	2301
6.476.3.8 listIterator . . . . .	2301
6.476.3.9 listIterator . . . . .	2301
6.476.3.10 remove . . . . .	2302
6.476.3.11 set . . . . .	2302
6.477 decaf::util::ListIterator< E > Class Template Reference . . . . .	2303
6.477.1 Detailed Description . . . . .	2304
6.477.2 Constructor & Destructor Documentation . . . . .	2304
6.477.2.1 ~ListIterator . . . . .	2304
6.477.3 Member Function Documentation . . . . .	2304
6.477.3.1 add . . . . .	2304
6.477.3.2 hasPrevious . . . . .	2305
6.477.3.3 nextIndex . . . . .	2305
6.477.3.4 previous . . . . .	2305
6.477.3.5 previousIndex . . . . .	2306
6.477.3.6 set . . . . .	2306
6.478 activemq::commands::LocalTransactionId Class Reference . . . . .	2306
6.478.1 Member Typedef Documentation . . . . .	2308
6.478.1.1 COMPARATOR . . . . .	2308
6.478.2 Constructor & Destructor Documentation . . . . .	2308
6.478.2.1 LocalTransactionId . . . . .	2308
6.478.2.2 LocalTransactionId . . . . .	2308
6.478.2.3 ~LocalTransactionId . . . . .	2308
6.478.3 Member Function Documentation . . . . .	2308
6.478.3.1 cloneDataStructure . . . . .	2308
6.478.3.2 compareTo . . . . .	2308
6.478.3.3 copyDataStructure . . . . .	2308

6.478.3.4 equals . . . . .	2309
6.478.3.5 equals . . . . .	2309
6.478.3.6 getConnectionId . . . . .	2309
6.478.3.7 getConnectionId . . . . .	2309
6.478.3.8 getDataStructureType . . . . .	2309
6.478.3.9 getValue . . . . .	2309
6.478.3.10operator< . . . . .	2309
6.478.3.11operator= . . . . .	2309
6.478.3.12operator== . . . . .	2309
6.478.3.13setConnectionId . . . . .	2310
6.478.3.14setValue . . . . .	2310
6.478.3.15toString . . . . .	2310
6.478.4 Field Documentation . . . . .	2310
6.478.4.1 connectionId . . . . .	2310
6.478.4.2 ID_LOCALTRANSACTIONID . . . . .	2310
6.478.4.3 value . . . . .	2310
6.479activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller	
Class Reference . . . . .	2310
6.479.1 Detailed Description . . . . .	2311
6.479.2 Constructor & Destructor Documentation . . . . .	2311
6.479.2.1 LocalTransactionIdMarshaller . . . . .	2311
6.479.2.2 ~LocalTransactionIdMarshaller . . . . .	2311
6.479.3 Member Function Documentation . . . . .	2311
6.479.3.1 createObject . . . . .	2311
6.479.3.2 getDataStructureType . . . . .	2312
6.479.3.3 looseMarshal . . . . .	2312
6.479.3.4 looseUnmarshal . . . . .	2312
6.479.3.5 tightMarshal1 . . . . .	2313
6.479.3.6 tightMarshal2 . . . . .	2313
6.479.3.7 tightUnmarshal . . . . .	2314
6.480activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	
Class Reference . . . . .	2314
6.480.1 Detailed Description . . . . .	2315
6.480.2 Constructor & Destructor Documentation . . . . .	2315

6.480.2.1 LocalTransactionIdMarshaller . . . . .	2315
6.480.2.2 ~LocalTransactionIdMarshaller . . . . .	2315
6.480.3 Member Function Documentation . . . . .	2315
6.480.3.1 createObject . . . . .	2315
6.480.3.2 getDataStructureType . . . . .	2316
6.480.3.3 looseMarshal . . . . .	2316
6.480.3.4 looseUnmarshal . . . . .	2316
6.480.3.5 tightMarshal1 . . . . .	2317
6.480.3.6 tightMarshal2 . . . . .	2317
6.480.3.7 tightUnmarshal . . . . .	2318
6.481 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	
Class Reference . . . . .	2318
6.481.1 Detailed Description . . . . .	2319
6.481.2 Constructor & Destructor Documentation . . . . .	2319
6.481.2.1 LocalTransactionIdMarshaller . . . . .	2319
6.481.2.2 ~LocalTransactionIdMarshaller . . . . .	2319
6.481.3 Member Function Documentation . . . . .	2319
6.481.3.1 createObject . . . . .	2319
6.481.3.2 getDataStructureType . . . . .	2320
6.481.3.3 looseMarshal . . . . .	2320
6.481.3.4 looseUnmarshal . . . . .	2320
6.481.3.5 tightMarshal1 . . . . .	2321
6.481.3.6 tightMarshal2 . . . . .	2321
6.481.3.7 tightUnmarshal . . . . .	2322
6.482 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller	
Class Reference . . . . .	2322
6.482.1 Detailed Description . . . . .	2323
6.482.2 Constructor & Destructor Documentation . . . . .	2323
6.482.2.1 LocalTransactionIdMarshaller . . . . .	2323
6.482.2.2 ~LocalTransactionIdMarshaller . . . . .	2323
6.482.3 Member Function Documentation . . . . .	2323
6.482.3.1 createObject . . . . .	2323
6.482.3.2 getDataStructureType . . . . .	2324
6.482.3.3 looseMarshal . . . . .	2324

6.482.3.4 looseUnmarshal . . . . .	2324
6.482.3.5 tightMarshal1 . . . . .	2325
6.482.3.6 tightMarshal2 . . . . .	2325
6.482.3.7 tightUnmarshal . . . . .	2326
6.483activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller	
Class Reference . . . . .	2326
6.483.1 Detailed Description . . . . .	2327
6.483.2 Constructor & Destructor Documentation . . . . .	2327
6.483.2.1 LocalTransactionIdMarshaller . . . . .	2327
6.483.2.2 ~LocalTransactionIdMarshaller . . . . .	2327
6.483.3 Member Function Documentation . . . . .	2327
6.483.3.1 createObject . . . . .	2327
6.483.3.2 getDataStructureType . . . . .	2328
6.483.3.3 looseMarshal . . . . .	2328
6.483.3.4 looseUnmarshal . . . . .	2328
6.483.3.5 tightMarshal1 . . . . .	2329
6.483.3.6 tightMarshal2 . . . . .	2329
6.483.3.7 tightUnmarshal . . . . .	2330
6.484activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	
Class Reference . . . . .	2330
6.484.1 Detailed Description . . . . .	2331
6.484.2 Constructor & Destructor Documentation . . . . .	2331
6.484.2.1 LocalTransactionIdMarshaller . . . . .	2331
6.484.2.2 ~LocalTransactionIdMarshaller . . . . .	2331
6.484.3 Member Function Documentation . . . . .	2331
6.484.3.1 createObject . . . . .	2331
6.484.3.2 getDataStructureType . . . . .	2332
6.484.3.3 looseMarshal . . . . .	2332
6.484.3.4 looseUnmarshal . . . . .	2332
6.484.3.5 tightMarshal1 . . . . .	2333
6.484.3.6 tightMarshal2 . . . . .	2333
6.484.3.7 tightUnmarshal . . . . .	2334
6.485decaf::util::concurrent::Lock Class Reference . . . . .	2334
6.485.1 Detailed Description . . . . .	2335

6.485.2 Constructor & Destructor Documentation . . . . .	2335
6.485.2.1 Lock . . . . .	2335
6.485.2.2 ~Lock . . . . .	2335
6.485.3 Member Function Documentation . . . . .	2335
6.485.3.1 isLocked . . . . .	2335
6.485.3.2 lock . . . . .	2335
6.485.3.3 unlock . . . . .	2336
6.486decaf::util::concurrent::locks::Lock Class Reference . . . . .	2336
6.486.1 Detailed Description . . . . .	2336
6.486.2 Constructor & Destructor Documentation . . . . .	2338
6.486.2.1 ~Lock . . . . .	2338
6.486.3 Member Function Documentation . . . . .	2338
6.486.3.1 lock . . . . .	2338
6.486.3.2 lockInterruptibly . . . . .	2338
6.486.3.3 newCondition . . . . .	2339
6.486.3.4 tryLock . . . . .	2339
6.486.3.5 tryLock . . . . .	2340
6.486.3.6 unlock . . . . .	2341
6.487decaf::util::concurrent::locks::LockSupport Class Reference . . . . .	2341
6.487.1 Detailed Description . . . . .	2342
6.487.2 Constructor & Destructor Documentation . . . . .	2343
6.487.2.1 ~LockSupport . . . . .	2343
6.487.3 Member Function Documentation . . . . .	2343
6.487.3.1 park . . . . .	2343
6.487.3.2 parkNanos . . . . .	2343
6.487.3.3 parkUntil . . . . .	2344
6.487.3.4 unpark . . . . .	2344
6.488decaf::util::logging::Logger Class Reference . . . . .	2345
6.488.1 Detailed Description . . . . .	2347
6.488.2 Constructor & Destructor Documentation . . . . .	2348
6.488.2.1 Logger . . . . .	2348
6.488.2.2 ~Logger . . . . .	2348
6.488.3 Member Function Documentation . . . . .	2348
6.488.3.1 addHandler . . . . .	2348

6.488.3.2 config . . . . .	2349
6.488.3.3 debug . . . . .	2349
6.488.3.4 entering . . . . .	2349
6.488.3.5 exiting . . . . .	2350
6.488.3.6 fine . . . . .	2350
6.488.3.7 finer . . . . .	2350
6.488.3.8 finest . . . . .	2351
6.488.3.9 getAnonymousLogger . . . . .	2351
6.488.3.10getFilter . . . . .	2351
6.488.3.11getHandlers . . . . .	2351
6.488.3.12getLevel . . . . .	2352
6.488.3.13getLogger . . . . .	2352
6.488.3.14getName . . . . .	2352
6.488.3.15getParent . . . . .	2352
6.488.3.16getUseParentHandlers . . . . .	2353
6.488.3.17info . . . . .	2353
6.488.3.18sLoggable . . . . .	2353
6.488.3.19log . . . . .	2354
6.488.3.20log . . . . .	2354
6.488.3.21log . . . . .	2354
6.488.3.22log . . . . .	2354
6.488.3.23removeHandler . . . . .	2355
6.488.3.24setFilter . . . . .	2355
6.488.3.25setLevel . . . . .	2355
6.488.3.26setParent . . . . .	2356
6.488.3.27setUseParentHandlers . . . . .	2356
6.488.3.28severe . . . . .	2356
6.488.3.29throwing . . . . .	2356
6.488.3.30warning . . . . .	2357
6.489decaf::util::logging::LoggerHierarchy Class Reference . . . . .	2357
6.489.1 Constructor & Destructor Documentation . . . . .	2357
6.489.1.1 LoggerHierarchy . . . . .	2357
6.489.1.2 ~LoggerHierarchy . . . . .	2357
6.490activemq::io::LoggingInputStream Class Reference . . . . .	2358



6.490.1 Constructor & Destructor Documentation . . . . .	2358
6.490.1.1 LoggingInputStream . . . . .	2358
6.490.1.2 ~LoggingInputStream . . . . .	2358
6.490.2 Member Function Documentation . . . . .	2358
6.490.2.1 doReadArrayBounded . . . . .	2359
6.490.2.2 doReadByte . . . . .	2359
6.491 activemq::io::LoggingOutputStream Class Reference . . . . .	2359
6.491.1 Detailed Description . . . . .	2359
6.491.2 Constructor & Destructor Documentation . . . . .	2360
6.491.2.1 LoggingOutputStream . . . . .	2360
6.491.2.2 ~LoggingOutputStream . . . . .	2360
6.491.3 Member Function Documentation . . . . .	2360
6.491.3.1 doWriteArrayBounded . . . . .	2360
6.491.3.2 doWriteByte . . . . .	2360
6.492 activemq::transport::logging::LoggingTransport Class Reference . . . . .	2360
6.492.1 Detailed Description . . . . .	2361
6.492.2 Constructor & Destructor Documentation . . . . .	2361
6.492.2.1 LoggingTransport . . . . .	2361
6.492.2.2 ~LoggingTransport . . . . .	2361
6.492.3 Member Function Documentation . . . . .	2361
6.492.3.1 onCommand . . . . .	2361
6.492.3.2 oneway . . . . .	2362
6.492.3.3 request . . . . .	2362
6.492.3.4 request . . . . .	2363
6.493 decaf::util::logging::LogManager Class Reference . . . . .	2363
6.493.1 Detailed Description . . . . .	2364
6.493.2 Constructor & Destructor Documentation . . . . .	2366
6.493.2.1 ~LogManager . . . . .	2366
6.493.2.2 LogManager . . . . .	2366
6.493.2.3 LogManager . . . . .	2366
6.493.3 Member Function Documentation . . . . .	2366
6.493.3.1 addLogger . . . . .	2366
6.493.3.2 addPropertyChangeListener . . . . .	2367
6.493.3.3 getLogger . . . . .	2367

6.493.3.4	getLoggerNames . . . . .	2367
6.493.3.5	getLogManager . . . . .	2368
6.493.3.6	getProperties . . . . .	2368
6.493.3.7	getProperty . . . . .	2368
6.493.3.8	operator= . . . . .	2368
6.493.3.9	readConfiguration . . . . .	2368
6.493.3.10	readConfiguration . . . . .	2369
6.493.3.11	removePropertyChangeListener . . . . .	2369
6.493.3.12	reset . . . . .	2369
6.493.3.13	setProperties . . . . .	2370
6.493.4	Friends And Related Function Documentation . . . . .	2370
6.493.4.1	decaf::lang::Runtime . . . . .	2370
6.494	decaf::util::logging::LogRecord Class Reference . . . . .	2370
6.494.1	Detailed Description . . . . .	2371
6.494.2	Constructor & Destructor Documentation . . . . .	2371
6.494.2.1	LogRecord . . . . .	2371
6.494.2.2	~LogRecord . . . . .	2371
6.494.3	Member Function Documentation . . . . .	2371
6.494.3.1	getLevel . . . . .	2371
6.494.3.2	getLoggerName . . . . .	2372
6.494.3.3	getMessage . . . . .	2372
6.494.3.4	getSourceFile . . . . .	2372
6.494.3.5	getSourceFunction . . . . .	2372
6.494.3.6	getSourceLine . . . . .	2372
6.494.3.7	getThrown . . . . .	2373
6.494.3.8	getTimestamp . . . . .	2373
6.494.3.9	getTreadId . . . . .	2373
6.494.3.10	setLevel . . . . .	2373
6.494.3.11	setLoggerName . . . . .	2373
6.494.3.12	setMessage . . . . .	2374
6.494.3.13	setSourceFile . . . . .	2374
6.494.3.14	setSourceFunction . . . . .	2374
6.494.3.15	setSourceLine . . . . .	2374
6.494.3.16	setThrown . . . . .	2374

6.494.3.17	setTimestamp	2375
6.494.3.18	setTreadId	2375
6.495	decaf::util::logging::LogWriter Class Reference	2375
6.495.1	Constructor & Destructor Documentation	2376
6.495.1.1	LogWriter	2376
6.495.1.2	~LogWriter	2376
6.495.2	Member Function Documentation	2376
6.495.2.1	destroy	2376
6.495.2.2	getInstance	2376
6.495.2.3	log	2376
6.495.2.4	log	2376
6.495.2.5	returnInstance	2377
6.496	decaf::lang::Long Class Reference	2377
6.496.1	Constructor & Destructor Documentation	2379
6.496.1.1	Long	2379
6.496.1.2	Long	2380
6.496.1.3	~Long	2380
6.496.2	Member Function Documentation	2380
6.496.2.1	bitCount	2380
6.496.2.2	byteValue	2380
6.496.2.3	compareTo	2380
6.496.2.4	compareTo	2381
6.496.2.5	decode	2381
6.496.2.6	doubleValue	2382
6.496.2.7	equals	2382
6.496.2.8	equals	2382
6.496.2.9	floatValue	2382
6.496.2.10	highestOneBit	2383
6.496.2.11	intValue	2383
6.496.2.12	longValue	2383
6.496.2.13	lowestOneBit	2383
6.496.2.14	numberOfLeadingZeros	2384
6.496.2.15	numberOfTrailingZeros	2384
6.496.2.16	operator<	2385

6.496.2.17operator< . . . . .	2385
6.496.2.18operator== . . . . .	2385
6.496.2.19operator== . . . . .	2386
6.496.2.20parseLong . . . . .	2386
6.496.2.21parseLong . . . . .	2386
6.496.2.22reverse . . . . .	2387
6.496.2.23reverseBytes . . . . .	2387
6.496.2.24rotateLeft . . . . .	2387
6.496.2.25rotateRight . . . . .	2388
6.496.2.26shortValue . . . . .	2388
6.496.2.27signum . . . . .	2388
6.496.2.28toBinaryString . . . . .	2389
6.496.2.29toHexString . . . . .	2389
6.496.2.30toOctalString . . . . .	2390
6.496.2.31toString . . . . .	2390
6.496.2.32toString . . . . .	2390
6.496.2.33toString . . . . .	2390
6.496.2.34valueOf . . . . .	2391
6.496.2.35valueOf . . . . .	2391
6.496.2.36valueOf . . . . .	2391
6.496.3 Field Documentation . . . . .	2392
6.496.3.1 MAX_VALUE . . . . .	2392
6.496.3.2 MIN_VALUE . . . . .	2392
6.496.3.3 SIZE . . . . .	2392
6.497decaf::internal::nio::LongArrayBuffer Class Reference . . . . .	2392
6.497.1 Constructor & Destructor Documentation . . . . .	2396
6.497.1.1 LongArrayBuffer . . . . .	2396
6.497.1.2 LongArrayBuffer . . . . .	2396
6.497.1.3 LongArrayBuffer . . . . .	2397
6.497.1.4 LongArrayBuffer . . . . .	2397
6.497.1.5 ~LongArrayBuffer . . . . .	2398
6.497.2 Member Function Documentation . . . . .	2398
6.497.2.1 array . . . . .	2398
6.497.2.2 arrayOffset . . . . .	2398

6.497.2.3 asReadOnlyBuffer . . . . .	2399
6.497.2.4 compact . . . . .	2399
6.497.2.5 duplicate . . . . .	2400
6.497.2.6 get . . . . .	2400
6.497.2.7 get . . . . .	2400
6.497.2.8 hasArray . . . . .	2401
6.497.2.9 isReadOnly . . . . .	2401
6.497.2.10put . . . . .	2401
6.497.2.11put . . . . .	2402
6.497.2.12setReadOnly . . . . .	2402
6.497.2.13slice . . . . .	2403
6.498decaf::nio::LongBuffer Class Reference . . . . .	2403
6.498.1 Detailed Description . . . . .	2405
6.498.2 Constructor & Destructor Documentation . . . . .	2405
6.498.2.1 LongBuffer . . . . .	2405
6.498.2.2 ~LongBuffer . . . . .	2406
6.498.3 Member Function Documentation . . . . .	2406
6.498.3.1 allocate . . . . .	2406
6.498.3.2 array . . . . .	2406
6.498.3.3 arrayOffset . . . . .	2407
6.498.3.4 asReadOnlyBuffer . . . . .	2407
6.498.3.5 compact . . . . .	2407
6.498.3.6 compareTo . . . . .	2408
6.498.3.7 duplicate . . . . .	2408
6.498.3.8 equals . . . . .	2408
6.498.3.9 get . . . . .	2408
6.498.3.10get . . . . .	2409
6.498.3.11get . . . . .	2409
6.498.3.12get . . . . .	2410
6.498.3.13hasArray . . . . .	2410
6.498.3.14operator< . . . . .	2411
6.498.3.15operator== . . . . .	2411
6.498.3.16put . . . . .	2411
6.498.3.17put . . . . .	2411

6.498.3.18put . . . . .	2412
6.498.3.19put . . . . .	2413
6.498.3.20put . . . . .	2413
6.498.3.21slice . . . . .	2414
6.498.3.22toString . . . . .	2414
6.498.3.23wrap . . . . .	2414
6.498.3.24wrap . . . . .	2415
6.499activemq::util::LongSequenceGenerator Class Reference . . . . .	2415
6.499.1 Detailed Description . . . . .	2416
6.499.2 Constructor & Destructor Documentation . . . . .	2416
6.499.2.1 LongSequenceGenerator . . . . .	2416
6.499.2.2 ~LongSequenceGenerator . . . . .	2416
6.499.3 Member Function Documentation . . . . .	2416
6.499.3.1 getLastSequenceId . . . . .	2416
6.499.3.2 getNextSequenceId . . . . .	2416
6.500decaf::net::MalformedURLException Class Reference . . . . .	2416
6.500.1 Constructor & Destructor Documentation . . . . .	2417
6.500.1.1 MalformedURLException . . . . .	2417
6.500.1.2 MalformedURLException . . . . .	2417
6.500.1.3 MalformedURLException . . . . .	2417
6.500.1.4 MalformedURLException . . . . .	2417
6.500.1.5 MalformedURLException . . . . .	2418
6.500.1.6 MalformedURLException . . . . .	2418
6.500.1.7 ~MalformedURLException . . . . .	2418
6.500.2 Member Function Documentation . . . . .	2418
6.500.2.1 clone . . . . .	2418
6.501decaf::util::Map< K, V, COMPARATOR > Class Template Reference . . . . .	2419
6.501.1 Detailed Description . . . . .	2420
6.501.2 Constructor & Destructor Documentation . . . . .	2420
6.501.2.1 Map . . . . .	2420
6.501.2.2 ~Map . . . . .	2420
6.501.3 Member Function Documentation . . . . .	2420
6.501.3.1 clear . . . . .	2421
6.501.3.2 containsKey . . . . .	2421

6.501.3.3 containsValue . . . . .	2422
6.501.3.4 copy . . . . .	2423
6.501.3.5 equals . . . . .	2423
6.501.3.6 get . . . . .	2424
6.501.3.7 get . . . . .	2425
6.501.3.8 isEmpty . . . . .	2426
6.501.3.9 keySet . . . . .	2426
6.501.3.10put . . . . .	2427
6.501.3.11putAll . . . . .	2428
6.501.3.12remove . . . . .	2429
6.501.3.13size . . . . .	2430
6.501.3.14values . . . . .	2430
6.502cms::MapMessage Class Reference . . . . .	2431
6.502.1 Detailed Description . . . . .	2433
6.502.2 Constructor & Destructor Documentation . . . . .	2434
6.502.2.1 ~MapMessage . . . . .	2434
6.502.3 Member Function Documentation . . . . .	2434
6.502.3.1 getBoolean . . . . .	2434
6.502.3.2 getByte . . . . .	2434
6.502.3.3 getBytes . . . . .	2435
6.502.3.4 getChar . . . . .	2435
6.502.3.5 getDouble . . . . .	2435
6.502.3.6 getFloat . . . . .	2436
6.502.3.7 getInt . . . . .	2436
6.502.3.8 getLong . . . . .	2437
6.502.3.9 getMapNames . . . . .	2437
6.502.3.10getShort . . . . .	2437
6.502.3.11getString . . . . .	2438
6.502.3.12itemExists . . . . .	2438
6.502.3.13setBoolean . . . . .	2439
6.502.3.14setByte . . . . .	2439
6.502.3.15setBytes . . . . .	2439
6.502.3.16setChar . . . . .	2440
6.502.3.17setDouble . . . . .	2440

6.502.3.18	setFloat . . . . .	2441
6.502.3.19	setInt . . . . .	2441
6.502.3.20	setLong . . . . .	2442
6.502.3.21	setShort . . . . .	2442
6.502.3.22	setString . . . . .	2442
6.503	decaf::util::logging::MarkBlockLogger Class Reference . . . . .	2443
6.503.1	Detailed Description . . . . .	2443
6.503.2	Constructor & Destructor Documentation . . . . .	2443
6.503.2.1	MarkBlockLogger . . . . .	2443
6.503.2.2	~MarkBlockLogger . . . . .	2444
6.504	activemq::wireformat::MarshalAware Class Reference . . . . .	2444
6.504.1	Constructor & Destructor Documentation . . . . .	2445
6.504.1.1	~MarshalAware . . . . .	2445
6.504.2	Member Function Documentation . . . . .	2445
6.504.2.1	afterMarshal . . . . .	2445
6.504.2.2	afterUnmarshal . . . . .	2445
6.504.2.3	beforeMarshal . . . . .	2445
6.504.2.4	beforeUnmarshal . . . . .	2445
6.504.2.5	getMarshaledForm . . . . .	2446
6.504.2.6	isMarshalAware . . . . .	2446
6.504.2.7	setMarshaledForm . . . . .	2446
6.505	activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference . . . . .	2447
6.505.1	Detailed Description . . . . .	2447
6.505.2	Constructor & Destructor Documentation . . . . .	2447
6.505.2.1	~MarshallerFactory . . . . .	2447
6.505.3	Member Function Documentation . . . . .	2447
6.505.3.1	configure . . . . .	2447
6.506	activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference . . . . .	2447
6.506.1	Detailed Description . . . . .	2448
6.506.2	Constructor & Destructor Documentation . . . . .	2448
6.506.2.1	~MarshallerFactory . . . . .	2448
6.506.3	Member Function Documentation . . . . .	2448



6.506.3.1 configure . . . . .	2448
6.507activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference . . . . .	2448
6.507.1 Detailed Description . . . . .	2448
6.507.2 Constructor & Destructor Documentation . . . . .	2448
6.507.2.1 ~MarshallerFactory . . . . .	2448
6.507.3 Member Function Documentation . . . . .	2449
6.507.3.1 configure . . . . .	2449
6.508activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference . . . . .	2449
6.508.1 Detailed Description . . . . .	2449
6.508.2 Constructor & Destructor Documentation . . . . .	2449
6.508.2.1 ~MarshallerFactory . . . . .	2449
6.508.3 Member Function Documentation . . . . .	2449
6.508.3.1 configure . . . . .	2449
6.509activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference . . . . .	2450
6.509.1 Detailed Description . . . . .	2450
6.509.2 Constructor & Destructor Documentation . . . . .	2450
6.509.2.1 ~MarshallerFactory . . . . .	2450
6.509.3 Member Function Documentation . . . . .	2450
6.509.3.1 configure . . . . .	2450
6.510activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference . . . . .	2450
6.510.1 Detailed Description . . . . .	2451
6.510.2 Constructor & Destructor Documentation . . . . .	2451
6.510.2.1 ~MarshallerFactory . . . . .	2451
6.510.3 Member Function Documentation . . . . .	2451
6.510.3.1 configure . . . . .	2451
6.511activemq::util::MarshallingSupport Class Reference . . . . .	2451
6.511.1 Constructor & Destructor Documentation . . . . .	2452
6.511.1.1 MarshallingSupport . . . . .	2452
6.511.1.2 ~MarshallingSupport . . . . .	2452
6.511.2 Member Function Documentation . . . . .	2452
6.511.2.1 asciiToModifiedUtf8 . . . . .	2452

6.511.2.2 modifiedUtf8ToAscii . . . . .	2453
6.511.2.3 readString16 . . . . .	2453
6.511.2.4 readString32 . . . . .	2454
6.511.2.5 writeString . . . . .	2454
6.511.2.6 writeString16 . . . . .	2454
6.511.2.7 writeString32 . . . . .	2455
6.512decaf::lang::Math Class Reference . . . . .	2455
6.512.1 Detailed Description . . . . .	2457
6.512.2 Constructor & Destructor Documentation . . . . .	2457
6.512.2.1 Math . . . . .	2457
6.512.2.2 ~Math . . . . .	2457
6.512.3 Member Function Documentation . . . . .	2457
6.512.3.1 abs . . . . .	2457
6.512.3.2 abs . . . . .	2458
6.512.3.3 abs . . . . .	2458
6.512.3.4 abs . . . . .	2458
6.512.3.5 ceil . . . . .	2459
6.512.3.6 floor . . . . .	2460
6.512.3.7 max . . . . .	2460
6.512.3.8 max . . . . .	2461
6.512.3.9 max . . . . .	2461
6.512.3.10max . . . . .	2461
6.512.3.11max . . . . .	2462
6.512.3.12min . . . . .	2462
6.512.3.13min . . . . .	2462
6.512.3.14min . . . . .	2463
6.512.3.15min . . . . .	2463
6.512.3.16min . . . . .	2463
6.512.3.17min . . . . .	2464
6.512.3.18pow . . . . .	2464
6.512.3.19random . . . . .	2465
6.512.3.20round . . . . .	2466
6.512.3.21round . . . . .	2466
6.512.3.22signum . . . . .	2466

6.512.3.23 signum . . . . .	2467
6.512.3.24 sqrt . . . . .	2468
6.512.3.25 toDegrees . . . . .	2471
6.512.3.26 toRadians . . . . .	2472
6.512.4 Field Documentation . . . . .	2472
6.512.4.1 E . . . . .	2472
6.512.4.2 PI . . . . .	2472
6.513 activemq::util::MemoryUsage Class Reference . . . . .	2472
6.513.1 Constructor & Destructor Documentation . . . . .	2473
6.513.1.1 MemoryUsage . . . . .	2473
6.513.1.2 MemoryUsage . . . . .	2473
6.513.1.3 ~MemoryUsage . . . . .	2473
6.513.2 Member Function Documentation . . . . .	2473
6.513.2.1 decreaseUsage . . . . .	2473
6.513.2.2 enqueueUsage . . . . .	2474
6.513.2.3 getLimit . . . . .	2474
6.513.2.4 getUsage . . . . .	2474
6.513.2.5 increaseUsage . . . . .	2474
6.513.2.6 isFull . . . . .	2474
6.513.2.7 setLimit . . . . .	2475
6.513.2.8 setUsage . . . . .	2475
6.513.2.9 waitForSpace . . . . .	2475
6.513.2.10 waitForSpace . . . . .	2475
6.514 activemq::commands::Message Class Reference . . . . .	2475
6.514.1 Constructor & Destructor Documentation . . . . .	2480
6.514.1.1 Message . . . . .	2480
6.514.1.2 ~Message . . . . .	2480
6.514.2 Member Function Documentation . . . . .	2480
6.514.2.1 afterUnmarshal . . . . .	2480
6.514.2.2 beforeMarshal . . . . .	2480
6.514.2.3 cloneDataStructure . . . . .	2480
6.514.2.4 copyDataStructure . . . . .	2481
6.514.2.5 equals . . . . .	2481
6.514.2.6 getAckHandler . . . . .	2482

6.514.2.7	getArrival . . . . .	2482
6.514.2.8	getBrokerInTime . . . . .	2482
6.514.2.9	getBrokerOutTime . . . . .	2482
6.514.2.10	getBrokerPath . . . . .	2482
6.514.2.11	getBrokerPath . . . . .	2482
6.514.2.12	getCluster . . . . .	2482
6.514.2.13	getCluster . . . . .	2482
6.514.2.14	getConnection . . . . .	2482
6.514.2.15	getContent . . . . .	2482
6.514.2.16	getContent . . . . .	2482
6.514.2.17	getCorrelationId . . . . .	2483
6.514.2.18	getCorrelationId . . . . .	2483
6.514.2.19	getDataStructure . . . . .	2483
6.514.2.20	getDataStructure . . . . .	2483
6.514.2.21	getDataStructureType . . . . .	2483
6.514.2.22	getDestination . . . . .	2483
6.514.2.23	getDestination . . . . .	2483
6.514.2.24	getExpiration . . . . .	2483
6.514.2.25	getGroupID . . . . .	2483
6.514.2.26	getGroupID . . . . .	2483
6.514.2.27	getGroupSequence . . . . .	2483
6.514.2.28	getMarshaledProperties . . . . .	2483
6.514.2.29	getMarshaledProperties . . . . .	2484
6.514.2.30	getMessageId . . . . .	2484
6.514.2.31	getMessageId . . . . .	2484
6.514.2.32	getMessageProperties . . . . .	2484
6.514.2.33	getMessageProperties . . . . .	2484
6.514.2.34	getOriginalDestination . . . . .	2484
6.514.2.35	getOriginalDestination . . . . .	2484
6.514.2.36	getOriginalTransactionId . . . . .	2484
6.514.2.37	getOriginalTransactionId . . . . .	2484
6.514.2.38	getPriority . . . . .	2484
6.514.2.39	getProducerId . . . . .	2484
6.514.2.40	getProducerId . . . . .	2484

6.514.2.41	getRedeliveryCounter . . . . .	2485
6.514.2.42	getReplyTo . . . . .	2485
6.514.2.43	getReplyTo . . . . .	2485
6.514.2.44	getSize . . . . .	2485
6.514.2.45	getTargetConsumerId . . . . .	2485
6.514.2.46	getTargetConsumerId . . . . .	2485
6.514.2.47	getTimestamp . . . . .	2485
6.514.2.48	getTransactionId . . . . .	2485
6.514.2.49	getTransactionId . . . . .	2485
6.514.2.50	getType . . . . .	2485
6.514.2.51	getType . . . . .	2485
6.514.2.52	getUserID . . . . .	2485
6.514.2.53	getUserID . . . . .	2485
6.514.2.54	isCompressed . . . . .	2486
6.514.2.55	isDroppable . . . . .	2486
6.514.2.56	isExpired . . . . .	2486
6.514.2.57	isMarshalAware . . . . .	2486
6.514.2.58	isMessage . . . . .	2486
6.514.2.59	isPersistent . . . . .	2486
6.514.2.60	isReadOnlyBody . . . . .	2486
6.514.2.61	isReadOnlyProperties . . . . .	2487
6.514.2.62	isRecievedByDFBridge . . . . .	2487
6.514.2.63	onSend . . . . .	2487
6.514.2.64	setAckHandler . . . . .	2487
6.514.2.65	setArrival . . . . .	2487
6.514.2.66	setBrokerInTime . . . . .	2487
6.514.2.67	setBrokerOutTime . . . . .	2487
6.514.2.68	setBrokerPath . . . . .	2488
6.514.2.69	setCluster . . . . .	2488
6.514.2.70	setCompressed . . . . .	2488
6.514.2.71	setConnection . . . . .	2488
6.514.2.72	setContent . . . . .	2488
6.514.2.73	setCorrelationId . . . . .	2488
6.514.2.74	setDataStructure . . . . .	2488

6.514.2.75	setDestination . . . . .	2488
6.514.2.76	setDroppable . . . . .	2488
6.514.2.77	setExpiration . . . . .	2488
6.514.2.78	setGroupID . . . . .	2488
6.514.2.79	setGroupSequence . . . . .	2488
6.514.2.80	setMarshaledProperties . . . . .	2488
6.514.2.81	setMessageId . . . . .	2489
6.514.2.82	setOriginalDestination . . . . .	2489
6.514.2.83	setOriginalTransactionId . . . . .	2489
6.514.2.84	setPersistent . . . . .	2489
6.514.2.85	setPriority . . . . .	2489
6.514.2.86	setProducerId . . . . .	2489
6.514.2.87	setReadOnlyBody . . . . .	2489
6.514.2.88	setReadOnlyProperties . . . . .	2489
6.514.2.89	setRecievedByDFBridge . . . . .	2489
6.514.2.90	setRedeliveryCounter . . . . .	2489
6.514.2.91	setReplyTo . . . . .	2489
6.514.2.92	setTargetConsumerId . . . . .	2490
6.514.2.93	setTimestamp . . . . .	2490
6.514.2.94	setTransactionId . . . . .	2490
6.514.2.95	setType . . . . .	2490
6.514.2.96	setUserID . . . . .	2490
6.514.2.97	toString . . . . .	2490
6.514.2.98	visit . . . . .	2490
6.514.3	Field Documentation . . . . .	2490
6.514.3.1	arrival . . . . .	2491
6.514.3.2	brokerInTime . . . . .	2491
6.514.3.3	brokerOutTime . . . . .	2491
6.514.3.4	brokerPath . . . . .	2491
6.514.3.5	cluster . . . . .	2491
6.514.3.6	compressed . . . . .	2491
6.514.3.7	connection . . . . .	2491
6.514.3.8	content . . . . .	2491
6.514.3.9	correlationId . . . . .	2491

6.514.3.10	dataStructure	2491
6.514.3.11	DEFAULT_MESSAGE_SIZE	2491
6.514.3.12	destination	2491
6.514.3.13	droppable	2491
6.514.3.14	expiration	2491
6.514.3.15	groupId	2491
6.514.3.16	groupSequence	2491
6.514.3.17	ID_MESSAGE	2491
6.514.3.18	marshalledProperties	2492
6.514.3.19	messageId	2492
6.514.3.20	originalDestination	2492
6.514.3.21	originalTransactionId	2492
6.514.3.22	persistent	2492
6.514.3.23	priority	2492
6.514.3.24	producerId	2492
6.514.3.25	recievedByDFBridge	2492
6.514.3.26	redeliveryCounter	2492
6.514.3.27	replyTo	2492
6.514.3.28	targetConsumerId	2492
6.514.3.29	timestamp	2492
6.514.3.30	transactionId	2492
6.514.3.31	type	2492
6.514.3.32	userId	2492
6.515	cms::Message Class Reference	2493
6.515.1	Detailed Description	2495
6.515.2	Constructor & Destructor Documentation	2497
6.515.2.1	~Message	2497
6.515.3	Member Function Documentation	2497
6.515.3.1	acknowledge	2497
6.515.3.2	clearBody	2497
6.515.3.3	clearProperties	2498
6.515.3.4	clone	2498
6.515.3.5	getBooleanProperty	2499
6.515.3.6	getByteProperty	2499

6.515.3.7	getCMSCorrelationID	2500
6.515.3.8	getCMSDeliveryMode	2500
6.515.3.9	getCMSDestination	2501
6.515.3.10	getCMSExpiration	2501
6.515.3.11	getCMSMessageID	2502
6.515.3.12	getCMSPriority	2503
6.515.3.13	getCMSRedelivered	2503
6.515.3.14	getCMSReplyTo	2504
6.515.3.15	getCMSTimestamp	2504
6.515.3.16	getCMSType	2505
6.515.3.17	getDoubleProperty	2506
6.515.3.18	getFloatProperty	2506
6.515.3.19	getIntProperty	2507
6.515.3.20	getLongProperty	2507
6.515.3.21	getPropertyNames	2508
6.515.3.22	getShortProperty	2509
6.515.3.23	getStringProperty	2509
6.515.3.24	propertyExists	2510
6.515.3.25	setBooleanProperty	2510
6.515.3.26	setByteProperty	2511
6.515.3.27	setCMSCorrelationID	2511
6.515.3.28	setCMSDeliveryMode	2512
6.515.3.29	setCMSDestination	2513
6.515.3.30	setCMSExpiration	2513
6.515.3.31	setCMSMessageID	2514
6.515.3.32	setCMSPriority	2514
6.515.3.33	setCMSRedelivered	2515
6.515.3.34	setCMSReplyTo	2515
6.515.3.35	setCMSTimestamp	2516
6.515.3.36	setCMSType	2516
6.515.3.37	setDoubleProperty	2517
6.515.3.38	setFloatProperty	2518
6.515.3.39	setIntProperty	2518
6.515.3.40	setLongProperty	2519



6.515.3.41	setShortProperty . . . . .	2519
6.515.3.42	setStringProperty . . . . .	2520
6.516	activemq::commands::MessageAck Class Reference . . . . .	2521
6.516.1	Constructor & Destructor Documentation . . . . .	2522
6.516.1.1	MessageAck . . . . .	2522
6.516.1.2	~MessageAck . . . . .	2522
6.516.2	Member Function Documentation . . . . .	2522
6.516.2.1	cloneDataStructure . . . . .	2522
6.516.2.2	copyDataStructure . . . . .	2522
6.516.2.3	equals . . . . .	2523
6.516.2.4	getAckType . . . . .	2523
6.516.2.5	getConsumerId . . . . .	2523
6.516.2.6	getConsumerId . . . . .	2523
6.516.2.7	getDataStructureType . . . . .	2523
6.516.2.8	getDestination . . . . .	2523
6.516.2.9	getDestination . . . . .	2523
6.516.2.10	getFirstMessageld . . . . .	2524
6.516.2.11	getFirstMessageld . . . . .	2524
6.516.2.12	getLastMessageld . . . . .	2524
6.516.2.13	getLastMessageld . . . . .	2524
6.516.2.14	getMessageCount . . . . .	2524
6.516.2.15	getTransactionId . . . . .	2524
6.516.2.16	getTransactionId . . . . .	2524
6.516.2.17	isMessageAck . . . . .	2524
6.516.2.18	setAckType . . . . .	2524
6.516.2.19	setConsumerId . . . . .	2524
6.516.2.20	setDestination . . . . .	2524
6.516.2.21	setFirstMessageld . . . . .	2524
6.516.2.22	setLastMessageld . . . . .	2525
6.516.2.23	setMessageCount . . . . .	2525
6.516.2.24	setTransactionId . . . . .	2525
6.516.2.25	toString . . . . .	2525
6.516.2.26	visit . . . . .	2525
6.516.3	Field Documentation . . . . .	2525

6.516.3.1 ackType . . . . .	2525
6.516.3.2 consumerId . . . . .	2525
6.516.3.3 destination . . . . .	2525
6.516.3.4 firstMessageId . . . . .	2526
6.516.3.5 ID_MESSAGEACK . . . . .	2526
6.516.3.6 lastMessageId . . . . .	2526
6.516.3.7 messageCount . . . . .	2526
6.516.3.8 transactionId . . . . .	2526
6.517activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller Class	
Reference . . . . .	2526
6.517.1 Detailed Description . . . . .	2527
6.517.2 Constructor & Destructor Documentation . . . . .	2527
6.517.2.1 MessageAckMarshaller . . . . .	2527
6.517.2.2 ~MessageAckMarshaller . . . . .	2527
6.517.3 Member Function Documentation . . . . .	2527
6.517.3.1 createObject . . . . .	2527
6.517.3.2 getDataStructureType . . . . .	2527
6.517.3.3 looseMarshal . . . . .	2528
6.517.3.4 looseUnmarshal . . . . .	2528
6.517.3.5 tightMarshal1 . . . . .	2529
6.517.3.6 tightMarshal2 . . . . .	2529
6.517.3.7 tightUnmarshal . . . . .	2530
6.518activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class	
Reference . . . . .	2530
6.518.1 Detailed Description . . . . .	2531
6.518.2 Constructor & Destructor Documentation . . . . .	2531
6.518.2.1 MessageAckMarshaller . . . . .	2531
6.518.2.2 ~MessageAckMarshaller . . . . .	2531
6.518.3 Member Function Documentation . . . . .	2531
6.518.3.1 createObject . . . . .	2531
6.518.3.2 getDataStructureType . . . . .	2532
6.518.3.3 looseMarshal . . . . .	2532
6.518.3.4 looseUnmarshal . . . . .	2532
6.518.3.5 tightMarshal1 . . . . .	2533

6.518.3.6	tightMarshal2 . . . . .	2533
6.518.3.7	tightUnmarshal . . . . .	2534
6.519	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class	
	Reference . . . . .	2534
6.519.1	Detailed Description . . . . .	2535
6.519.2	Constructor & Destructor Documentation . . . . .	2535
6.519.2.1	MessageAckMarshaller . . . . .	2535
6.519.2.2	~MessageAckMarshaller . . . . .	2535
6.519.3	Member Function Documentation . . . . .	2535
6.519.3.1	createObject . . . . .	2535
6.519.3.2	getDataStructureType . . . . .	2536
6.519.3.3	looseMarshal . . . . .	2536
6.519.3.4	looseUnmarshal . . . . .	2536
6.519.3.5	tightMarshal1 . . . . .	2537
6.519.3.6	tightMarshal2 . . . . .	2537
6.519.3.7	tightUnmarshal . . . . .	2538
6.520	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class	
	Reference . . . . .	2538
6.520.1	Detailed Description . . . . .	2539
6.520.2	Constructor & Destructor Documentation . . . . .	2539
6.520.2.1	MessageAckMarshaller . . . . .	2539
6.520.2.2	~MessageAckMarshaller . . . . .	2539
6.520.3	Member Function Documentation . . . . .	2539
6.520.3.1	createObject . . . . .	2539
6.520.3.2	getDataStructureType . . . . .	2540
6.520.3.3	looseMarshal . . . . .	2540
6.520.3.4	looseUnmarshal . . . . .	2540
6.520.3.5	tightMarshal1 . . . . .	2541
6.520.3.6	tightMarshal2 . . . . .	2541
6.520.3.7	tightUnmarshal . . . . .	2542
6.521	activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class	
	Reference . . . . .	2542
6.521.1	Detailed Description . . . . .	2543
6.521.2	Constructor & Destructor Documentation . . . . .	2543
6.521.2.1	MessageAckMarshaller . . . . .	2543

6.521.2.2 ~MessageAckMarshaller . . . . .	2543
6.521.3 Member Function Documentation . . . . .	2543
6.521.3.1 createObject . . . . .	2543
6.521.3.2 getDataStructureType . . . . .	2544
6.521.3.3 looseMarshal . . . . .	2544
6.521.3.4 looseUnmarshal . . . . .	2544
6.521.3.5 tightMarshal1 . . . . .	2545
6.521.3.6 tightMarshal2 . . . . .	2545
6.521.3.7 tightUnmarshal . . . . .	2546
6.522activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class	
Reference . . . . .	2546
6.522.1 Detailed Description . . . . .	2547
6.522.2 Constructor & Destructor Documentation . . . . .	2547
6.522.2.1 MessageAckMarshaller . . . . .	2547
6.522.2.2 ~MessageAckMarshaller . . . . .	2547
6.522.3 Member Function Documentation . . . . .	2547
6.522.3.1 createObject . . . . .	2547
6.522.3.2 getDataStructureType . . . . .	2548
6.522.3.3 looseMarshal . . . . .	2548
6.522.3.4 looseUnmarshal . . . . .	2548
6.522.3.5 tightMarshal1 . . . . .	2549
6.522.3.6 tightMarshal2 . . . . .	2549
6.522.3.7 tightUnmarshal . . . . .	2550
6.523cms::MessageConsumer Class Reference . . . . .	2550
6.523.1 Detailed Description . . . . .	2551
6.523.2 Constructor & Destructor Documentation . . . . .	2551
6.523.2.1 ~MessageConsumer . . . . .	2551
6.523.3 Member Function Documentation . . . . .	2551
6.523.3.1 getMessageListener . . . . .	2552
6.523.3.2 getMessageSelector . . . . .	2552
6.523.3.3 receive . . . . .	2552
6.523.3.4 receive . . . . .	2553
6.523.3.5 receiveNoWait . . . . .	2553
6.523.3.6 setMessageListener . . . . .	2553

6.524	activemq::cmsutil::MessageCreator Class Reference . . . . .	2554
6.524.1	Detailed Description . . . . .	2554
6.524.2	Constructor & Destructor Documentation . . . . .	2554
6.524.2.1	~MessageCreator . . . . .	2554
6.524.3	Member Function Documentation . . . . .	2554
6.524.3.1	createMessage . . . . .	2554
6.525	activemq::commands::MessageDispatch Class Reference . . . . .	2555
6.525.1	Constructor & Destructor Documentation . . . . .	2556
6.525.1.1	MessageDispatch . . . . .	2556
6.525.1.2	~MessageDispatch . . . . .	2556
6.525.2	Member Function Documentation . . . . .	2556
6.525.2.1	cloneDataStructure . . . . .	2556
6.525.2.2	copyDataStructure . . . . .	2556
6.525.2.3	equals . . . . .	2557
6.525.2.4	getConsumerId . . . . .	2557
6.525.2.5	getConsumerId . . . . .	2557
6.525.2.6	getDataStructureType . . . . .	2557
6.525.2.7	getDestination . . . . .	2557
6.525.2.8	getDestination . . . . .	2557
6.525.2.9	getMessage . . . . .	2557
6.525.2.10	getMessage . . . . .	2558
6.525.2.11	getRedeliveryCounter . . . . .	2558
6.525.2.12	isMessageDispatch . . . . .	2558
6.525.2.13	setConsumerId . . . . .	2558
6.525.2.14	setDestination . . . . .	2558
6.525.2.15	setMessage . . . . .	2558
6.525.2.16	setRedeliveryCounter . . . . .	2558
6.525.2.17	toString . . . . .	2558
6.525.2.18	visit . . . . .	2558
6.525.3	Field Documentation . . . . .	2559
6.525.3.1	consumerId . . . . .	2559
6.525.3.2	destination . . . . .	2559
6.525.3.3	ID_MESSAGEDISPATCH . . . . .	2559
6.525.3.4	message . . . . .	2559

6.525.3.5 redeliveryCounter . . . . .	2559
6.526activemq::core::MessageDispatchChannel Class Reference . . . . .	2559
6.526.1 Constructor & Destructor Documentation . . . . .	2561
6.526.1.1 MessageDispatchChannel . . . . .	2561
6.526.1.2 ~MessageDispatchChannel . . . . .	2561
6.526.2 Member Function Documentation . . . . .	2561
6.526.2.1 clear . . . . .	2561
6.526.2.2 close . . . . .	2561
6.526.2.3 dequeue . . . . .	2561
6.526.2.4 dequeueNoWait . . . . .	2561
6.526.2.5 enqueue . . . . .	2562
6.526.2.6 enqueueFirst . . . . .	2562
6.526.2.7 isClosed . . . . .	2562
6.526.2.8 isEmpty . . . . .	2562
6.526.2.9 isRunning . . . . .	2562
6.526.2.10lock . . . . .	2562
6.526.2.11notify . . . . .	2563
6.526.2.12notifyAll . . . . .	2563
6.526.2.13peek . . . . .	2563
6.526.2.14removeAll . . . . .	2564
6.526.2.15size . . . . .	2564
6.526.2.16start . . . . .	2564
6.526.2.17stop . . . . .	2564
6.526.2.18tryLock . . . . .	2564
6.526.2.19unlock . . . . .	2564
6.526.2.20wait . . . . .	2565
6.526.2.21wait . . . . .	2565
6.526.2.22wait . . . . .	2566
6.527activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller Class Reference . . . . .	2566
6.527.1 Detailed Description . . . . .	2567
6.527.2 Constructor & Destructor Documentation . . . . .	2567
6.527.2.1 MessageDispatchMarshaller . . . . .	2567
6.527.2.2 ~MessageDispatchMarshaller . . . . .	2567

6.527.3 Member Function Documentation . . . . .	2567
6.527.3.1 createObject . . . . .	2567
6.527.3.2 getDataStructureType . . . . .	2568
6.527.3.3 looseMarshal . . . . .	2568
6.527.3.4 looseUnmarshal . . . . .	2568
6.527.3.5 tightMarshal1 . . . . .	2569
6.527.3.6 tightMarshal2 . . . . .	2569
6.527.3.7 tightUnmarshal . . . . .	2570
6.528activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	
Class Reference . . . . .	2570
6.528.1 Detailed Description . . . . .	2571
6.528.2 Constructor & Destructor Documentation . . . . .	2571
6.528.2.1 MessageDispatchMarshaller . . . . .	2571
6.528.2.2 ~MessageDispatchMarshaller . . . . .	2571
6.528.3 Member Function Documentation . . . . .	2571
6.528.3.1 createObject . . . . .	2571
6.528.3.2 getDataStructureType . . . . .	2572
6.528.3.3 looseMarshal . . . . .	2572
6.528.3.4 looseUnmarshal . . . . .	2572
6.528.3.5 tightMarshal1 . . . . .	2573
6.528.3.6 tightMarshal2 . . . . .	2573
6.528.3.7 tightUnmarshal . . . . .	2574
6.529activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	
Class Reference . . . . .	2574
6.529.1 Detailed Description . . . . .	2575
6.529.2 Constructor & Destructor Documentation . . . . .	2575
6.529.2.1 MessageDispatchMarshaller . . . . .	2575
6.529.2.2 ~MessageDispatchMarshaller . . . . .	2575
6.529.3 Member Function Documentation . . . . .	2575
6.529.3.1 createObject . . . . .	2575
6.529.3.2 getDataStructureType . . . . .	2576
6.529.3.3 looseMarshal . . . . .	2576
6.529.3.4 looseUnmarshal . . . . .	2576
6.529.3.5 tightMarshal1 . . . . .	2577

6.529.3.6 tightMarshal2 . . . . .	2577
6.529.3.7 tightUnmarshal . . . . .	2578
6.530activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	
Class Reference . . . . .	2578
6.530.1 Detailed Description . . . . .	2579
6.530.2 Constructor & Destructor Documentation . . . . .	2579
6.530.2.1 MessageDispatchMarshaller . . . . .	2579
6.530.2.2 ~MessageDispatchMarshaller . . . . .	2579
6.530.3 Member Function Documentation . . . . .	2579
6.530.3.1 createObject . . . . .	2579
6.530.3.2 getDataStructureType . . . . .	2580
6.530.3.3 looseMarshal . . . . .	2580
6.530.3.4 looseUnmarshal . . . . .	2580
6.530.3.5 tightMarshal1 . . . . .	2581
6.530.3.6 tightMarshal2 . . . . .	2581
6.530.3.7 tightUnmarshal . . . . .	2582
6.531activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	
Class Reference . . . . .	2582
6.531.1 Detailed Description . . . . .	2583
6.531.2 Constructor & Destructor Documentation . . . . .	2583
6.531.2.1 MessageDispatchMarshaller . . . . .	2583
6.531.2.2 ~MessageDispatchMarshaller . . . . .	2583
6.531.3 Member Function Documentation . . . . .	2583
6.531.3.1 createObject . . . . .	2583
6.531.3.2 getDataStructureType . . . . .	2584
6.531.3.3 looseMarshal . . . . .	2584
6.531.3.4 looseUnmarshal . . . . .	2584
6.531.3.5 tightMarshal1 . . . . .	2585
6.531.3.6 tightMarshal2 . . . . .	2585
6.531.3.7 tightUnmarshal . . . . .	2586
6.532activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller	
Class Reference . . . . .	2586
6.532.1 Detailed Description . . . . .	2587
6.532.2 Constructor & Destructor Documentation . . . . .	2587
6.532.2.1 MessageDispatchMarshaller . . . . .	2587



6.532.2.2 ~MessageDispatchMarshaller . . . . .	2587
6.532.3 Member Function Documentation . . . . .	2587
6.532.3.1 createObject . . . . .	2587
6.532.3.2 getDataStructureType . . . . .	2588
6.532.3.3 looseMarshal . . . . .	2588
6.532.3.4 looseUnmarshal . . . . .	2588
6.532.3.5 tightMarshal1 . . . . .	2589
6.532.3.6 tightMarshal2 . . . . .	2589
6.532.3.7 tightUnmarshal . . . . .	2590
6.533activemq::commands::MessageDispatchNotification Class Reference . .	2590
6.533.1 Constructor & Destructor Documentation . . . . .	2592
6.533.1.1 MessageDispatchNotification . . . . .	2592
6.533.1.2 ~MessageDispatchNotification . . . . .	2592
6.533.2 Member Function Documentation . . . . .	2592
6.533.2.1 cloneDataStructure . . . . .	2592
6.533.2.2 copyDataStructure . . . . .	2592
6.533.2.3 equals . . . . .	2592
6.533.2.4 getConsumerId . . . . .	2593
6.533.2.5 getConsumerId . . . . .	2593
6.533.2.6 getDataStructureType . . . . .	2593
6.533.2.7 getDeliverySequenceId . . . . .	2593
6.533.2.8 getDestination . . . . .	2593
6.533.2.9 getDestination . . . . .	2593
6.533.2.10getMessageId . . . . .	2593
6.533.2.11getMessageId . . . . .	2593
6.533.2.12sMessageDispatchNotification . . . . .	2593
6.533.2.13setConsumerId . . . . .	2594
6.533.2.14setDeliverySequenceId . . . . .	2594
6.533.2.15setDestination . . . . .	2594
6.533.2.16setMessageId . . . . .	2594
6.533.2.17toString . . . . .	2594
6.533.2.18visit . . . . .	2594
6.533.3 Field Documentation . . . . .	2594
6.533.3.1 consumerId . . . . .	2594

6.533.3.2 deliverySequenceId . . . . .	2594
6.533.3.3 destination . . . . .	2595
6.533.3.4 ID_MESSAGEDISPATCHNOTIFICATION . . . . .	2595
6.533.3.5 messageId . . . . .	2595
6.534activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2595
6.534.1 Detailed Description . . . . .	2596
6.534.2 Constructor & Destructor Documentation . . . . .	2596
6.534.2.1 MessageDispatchNotificationMarshaller . . . . .	2596
6.534.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2596
6.534.3 Member Function Documentation . . . . .	2596
6.534.3.1 createObject . . . . .	2596
6.534.3.2 getDataStructureType . . . . .	2596
6.534.3.3 looseMarshal . . . . .	2597
6.534.3.4 looseUnmarshal . . . . .	2597
6.534.3.5 tightMarshal1 . . . . .	2597
6.534.3.6 tightMarshal2 . . . . .	2598
6.534.3.7 tightUnmarshal . . . . .	2598
6.535activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2599
6.535.1 Detailed Description . . . . .	2600
6.535.2 Constructor & Destructor Documentation . . . . .	2600
6.535.2.1 MessageDispatchNotificationMarshaller . . . . .	2600
6.535.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2600
6.535.3 Member Function Documentation . . . . .	2600
6.535.3.1 createObject . . . . .	2600
6.535.3.2 getDataStructureType . . . . .	2600
6.535.3.3 looseMarshal . . . . .	2601
6.535.3.4 looseUnmarshal . . . . .	2601
6.535.3.5 tightMarshal1 . . . . .	2602
6.535.3.6 tightMarshal2 . . . . .	2602
6.535.3.7 tightUnmarshal . . . . .	2603
6.536activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2603
6.536.1 Detailed Description . . . . .	2604

6.536.2 Constructor & Destructor Documentation . . . . .	2604
6.536.2.1 MessageDispatchNotificationMarshaller . . . . .	2604
6.536.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2604
6.536.3 Member Function Documentation . . . . .	2604
6.536.3.1 createObject . . . . .	2604
6.536.3.2 getDataStructureType . . . . .	2605
6.536.3.3 looseMarshal . . . . .	2605
6.536.3.4 looseUnmarshal . . . . .	2605
6.536.3.5 tightMarshal1 . . . . .	2606
6.536.3.6 tightMarshal2 . . . . .	2606
6.536.3.7 tightUnmarshal . . . . .	2607
6.537activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2607
6.537.1 Detailed Description . . . . .	2608
6.537.2 Constructor & Destructor Documentation . . . . .	2608
6.537.2.1 MessageDispatchNotificationMarshaller . . . . .	2608
6.537.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2608
6.537.3 Member Function Documentation . . . . .	2608
6.537.3.1 createObject . . . . .	2608
6.537.3.2 getDataStructureType . . . . .	2609
6.537.3.3 looseMarshal . . . . .	2609
6.537.3.4 looseUnmarshal . . . . .	2609
6.537.3.5 tightMarshal1 . . . . .	2610
6.537.3.6 tightMarshal2 . . . . .	2610
6.537.3.7 tightUnmarshal . . . . .	2611
6.538activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	
Class Reference . . . . .	2611
6.538.1 Detailed Description . . . . .	2612
6.538.2 Constructor & Destructor Documentation . . . . .	2612
6.538.2.1 MessageDispatchNotificationMarshaller . . . . .	2612
6.538.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2612
6.538.3 Member Function Documentation . . . . .	2612
6.538.3.1 createObject . . . . .	2613
6.538.3.2 getDataStructureType . . . . .	2613

6.538.3.3 looseMarshal . . . . .	2613
6.538.3.4 looseUnmarshal . . . . .	2614
6.538.3.5 tightMarshal1 . . . . .	2614
6.538.3.6 tightMarshal2 . . . . .	2615
6.538.3.7 tightUnmarshal . . . . .	2615
6.539activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller Class Reference . . . . .	2616
6.539.1 Detailed Description . . . . .	2616
6.539.2 Constructor & Destructor Documentation . . . . .	2617
6.539.2.1 MessageDispatchNotificationMarshaller . . . . .	2617
6.539.2.2 ~MessageDispatchNotificationMarshaller . . . . .	2617
6.539.3 Member Function Documentation . . . . .	2617
6.539.3.1 createObject . . . . .	2617
6.539.3.2 getDataStructureType . . . . .	2617
6.539.3.3 looseMarshal . . . . .	2617
6.539.3.4 looseUnmarshal . . . . .	2618
6.539.3.5 tightMarshal1 . . . . .	2618
6.539.3.6 tightMarshal2 . . . . .	2619
6.539.3.7 tightUnmarshal . . . . .	2619
6.540cms::MessageEnumeration Class Reference . . . . .	2620
6.540.1 Detailed Description . . . . .	2620
6.540.2 Constructor & Destructor Documentation . . . . .	2620
6.540.2.1 ~MessageEnumeration . . . . .	2620
6.540.3 Member Function Documentation . . . . .	2620
6.540.3.1 hasMoreMessages . . . . .	2620
6.540.3.2 nextMessage . . . . .	2621
6.541cms::MessageEOFException Class Reference . . . . .	2621
6.541.1 Detailed Description . . . . .	2622
6.541.2 Constructor & Destructor Documentation . . . . .	2622
6.541.2.1 MessageEOFException . . . . .	2622
6.541.2.2 MessageEOFException . . . . .	2622
6.541.2.3 MessageEOFException . . . . .	2622
6.541.2.4 MessageEOFException . . . . .	2622
6.541.2.5 ~MessageEOFException . . . . .	2622

6.542cms::MessageFormatException Class Reference . . . . .	2622
6.542.1 Detailed Description . . . . .	2623
6.542.2 Constructor & Destructor Documentation . . . . .	2623
6.542.2.1 MessageFormatException . . . . .	2623
6.542.2.2 MessageFormatException . . . . .	2623
6.542.2.3 MessageFormatException . . . . .	2623
6.542.2.4 MessageFormatException . . . . .	2623
6.542.2.5 ~MessageFormatException . . . . .	2623
6.543activemq::commands::MessageId Class Reference . . . . .	2623
6.543.1 Member Typedef Documentation . . . . .	2625
6.543.1.1 COMPARATOR . . . . .	2625
6.543.2 Constructor & Destructor Documentation . . . . .	2625
6.543.2.1 MessageId . . . . .	2625
6.543.2.2 MessageId . . . . .	2625
6.543.2.3 MessageId . . . . .	2625
6.543.2.4 MessageId . . . . .	2625
6.543.2.5 MessageId . . . . .	2625
6.543.2.6 MessageId . . . . .	2625
6.543.2.7 ~MessageId . . . . .	2625
6.543.3 Member Function Documentation . . . . .	2625
6.543.3.1 cloneDataStructure . . . . .	2625
6.543.3.2 compareTo . . . . .	2626
6.543.3.3 copyDataStructure . . . . .	2626
6.543.3.4 equals . . . . .	2626
6.543.3.5 equals . . . . .	2626
6.543.3.6 getBrokerSequenceId . . . . .	2626
6.543.3.7 getDataStructureType . . . . .	2626
6.543.3.8 getProducerId . . . . .	2627
6.543.3.9 getProducerId . . . . .	2627
6.543.3.10getProducerSequenceId . . . . .	2627
6.543.3.11operator< . . . . .	2627
6.543.3.12operator= . . . . .	2627
6.543.3.13operator== . . . . .	2627
6.543.3.14setBrokerSequenceId . . . . .	2627

6.543.3.15	setProducerId . . . . .	2627
6.543.3.16	setProducerSequenceId . . . . .	2627
6.543.3.17	setTextView . . . . .	2627
6.543.3.18	setValue . . . . .	2627
6.543.3.19	toString . . . . .	2627
6.543.4	Field Documentation . . . . .	2627
6.543.4.1	brokerSequenceId . . . . .	2627
6.543.4.2	ID_MESSAGEID . . . . .	2628
6.543.4.3	producerId . . . . .	2628
6.543.4.4	producerSequenceId . . . . .	2628
6.544	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class	
	Reference . . . . .	2628
6.544.1	Detailed Description . . . . .	2629
6.544.2	Constructor & Destructor Documentation . . . . .	2629
6.544.2.1	MessageIdMarshaller . . . . .	2629
6.544.2.2	~MessageIdMarshaller . . . . .	2629
6.544.3	Member Function Documentation . . . . .	2629
6.544.3.1	createObject . . . . .	2629
6.544.3.2	getDataStructureType . . . . .	2629
6.544.3.3	looseMarshal . . . . .	2630
6.544.3.4	looseUnmarshal . . . . .	2630
6.544.3.5	tightMarshal1 . . . . .	2630
6.544.3.6	tightMarshal2 . . . . .	2631
6.544.3.7	tightUnmarshal . . . . .	2631
6.545	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class	
	Reference . . . . .	2632
6.545.1	Detailed Description . . . . .	2633
6.545.2	Constructor & Destructor Documentation . . . . .	2633
6.545.2.1	MessageIdMarshaller . . . . .	2633
6.545.2.2	~MessageIdMarshaller . . . . .	2633
6.545.3	Member Function Documentation . . . . .	2633
6.545.3.1	createObject . . . . .	2633
6.545.3.2	getDataStructureType . . . . .	2633
6.545.3.3	looseMarshal . . . . .	2634

6.545.3.4 looseUnmarshal . . . . .	2634
6.545.3.5 tightMarshal1 . . . . .	2634
6.545.3.6 tightMarshal2 . . . . .	2635
6.545.3.7 tightUnmarshal . . . . .	2635
6.546activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class	
Reference . . . . .	2636
6.546.1 Detailed Description . . . . .	2637
6.546.2 Constructor & Destructor Documentation . . . . .	2637
6.546.2.1 MessageIdMarshaller . . . . .	2637
6.546.2.2 ~MessageIdMarshaller . . . . .	2637
6.546.3 Member Function Documentation . . . . .	2637
6.546.3.1 createObject . . . . .	2637
6.546.3.2 getDataStructureType . . . . .	2637
6.546.3.3 looseMarshal . . . . .	2638
6.546.3.4 looseUnmarshal . . . . .	2638
6.546.3.5 tightMarshal1 . . . . .	2638
6.546.3.6 tightMarshal2 . . . . .	2639
6.546.3.7 tightUnmarshal . . . . .	2639
6.547activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class	
Reference . . . . .	2640
6.547.1 Detailed Description . . . . .	2641
6.547.2 Constructor & Destructor Documentation . . . . .	2641
6.547.2.1 MessageIdMarshaller . . . . .	2641
6.547.2.2 ~MessageIdMarshaller . . . . .	2641
6.547.3 Member Function Documentation . . . . .	2641
6.547.3.1 createObject . . . . .	2641
6.547.3.2 getDataStructureType . . . . .	2641
6.547.3.3 looseMarshal . . . . .	2642
6.547.3.4 looseUnmarshal . . . . .	2642
6.547.3.5 tightMarshal1 . . . . .	2642
6.547.3.6 tightMarshal2 . . . . .	2643
6.547.3.7 tightUnmarshal . . . . .	2643
6.548activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class	
Reference . . . . .	2644
6.548.1 Detailed Description . . . . .	2645

6.548.2 Constructor & Destructor Documentation . . . . .	2645
6.548.2.1 MessageIdMarshaller . . . . .	2645
6.548.2.2 ~MessageIdMarshaller . . . . .	2645
6.548.3 Member Function Documentation . . . . .	2645
6.548.3.1 createObject . . . . .	2645
6.548.3.2 getDataStructureType . . . . .	2645
6.548.3.3 looseMarshal . . . . .	2646
6.548.3.4 looseUnmarshal . . . . .	2646
6.548.3.5 tightMarshal1 . . . . .	2646
6.548.3.6 tightMarshal2 . . . . .	2647
6.548.3.7 tightUnmarshal . . . . .	2647
6.549activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference . . . . .	2648
6.549.1 Detailed Description . . . . .	2649
6.549.2 Constructor & Destructor Documentation . . . . .	2649
6.549.2.1 MessageIdMarshaller . . . . .	2649
6.549.2.2 ~MessageIdMarshaller . . . . .	2649
6.549.3 Member Function Documentation . . . . .	2649
6.549.3.1 createObject . . . . .	2649
6.549.3.2 getDataStructureType . . . . .	2649
6.549.3.3 looseMarshal . . . . .	2650
6.549.3.4 looseUnmarshal . . . . .	2650
6.549.3.5 tightMarshal1 . . . . .	2650
6.549.3.6 tightMarshal2 . . . . .	2651
6.549.3.7 tightUnmarshal . . . . .	2651
6.550cms::MessageListener Class Reference . . . . .	2652
6.550.1 Detailed Description . . . . .	2652
6.550.2 Constructor & Destructor Documentation . . . . .	2652
6.550.2.1 ~MessageListener . . . . .	2652
6.550.3 Member Function Documentation . . . . .	2652
6.550.3.1 onMessage . . . . .	2652
6.551activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference . . . . .	2653
6.551.1 Detailed Description . . . . .	2654



6.551.2 Constructor & Destructor Documentation . . . . .	2654
6.551.2.1 MessageMarshaller . . . . .	2654
6.551.2.2 ~MessageMarshaller . . . . .	2654
6.551.3 Member Function Documentation . . . . .	2654
6.551.3.1 looseMarshal . . . . .	2654
6.551.3.2 looseUnmarshal . . . . .	2655
6.551.3.3 tightMarshal1 . . . . .	2655
6.551.3.4 tightMarshal2 . . . . .	2656
6.551.3.5 tightUnmarshal . . . . .	2656
6.552activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class	
Reference . . . . .	2657
6.552.1 Detailed Description . . . . .	2658
6.552.2 Constructor & Destructor Documentation . . . . .	2658
6.552.2.1 MessageMarshaller . . . . .	2658
6.552.2.2 ~MessageMarshaller . . . . .	2658
6.552.3 Member Function Documentation . . . . .	2658
6.552.3.1 looseMarshal . . . . .	2658
6.552.3.2 looseUnmarshal . . . . .	2659
6.552.3.3 tightMarshal1 . . . . .	2659
6.552.3.4 tightMarshal2 . . . . .	2660
6.552.3.5 tightUnmarshal . . . . .	2661
6.553activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class	
Reference . . . . .	2661
6.553.1 Detailed Description . . . . .	2662
6.553.2 Constructor & Destructor Documentation . . . . .	2662
6.553.2.1 MessageMarshaller . . . . .	2662
6.553.2.2 ~MessageMarshaller . . . . .	2662
6.553.3 Member Function Documentation . . . . .	2662
6.553.3.1 looseMarshal . . . . .	2663
6.553.3.2 looseUnmarshal . . . . .	2663
6.553.3.3 tightMarshal1 . . . . .	2664
6.553.3.4 tightMarshal2 . . . . .	2664
6.553.3.5 tightUnmarshal . . . . .	2665
6.554activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class	
Reference . . . . .	2666

6.554.1 Detailed Description . . . . .	2667
6.554.2 Constructor & Destructor Documentation . . . . .	2667
6.554.2.1 MessageMarshaller . . . . .	2667
6.554.2.2 ~MessageMarshaller . . . . .	2667
6.554.3 Member Function Documentation . . . . .	2667
6.554.3.1 looseMarshal . . . . .	2667
6.554.3.2 looseUnmarshal . . . . .	2668
6.554.3.3 tightMarshal1 . . . . .	2668
6.554.3.4 tightMarshal2 . . . . .	2669
6.554.3.5 tightUnmarshal . . . . .	2669
6.555activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class	
Reference . . . . .	2670
6.555.1 Detailed Description . . . . .	2671
6.555.2 Constructor & Destructor Documentation . . . . .	2671
6.555.2.1 MessageMarshaller . . . . .	2671
6.555.2.2 ~MessageMarshaller . . . . .	2671
6.555.3 Member Function Documentation . . . . .	2671
6.555.3.1 looseMarshal . . . . .	2671
6.555.3.2 looseUnmarshal . . . . .	2672
6.555.3.3 tightMarshal1 . . . . .	2672
6.555.3.4 tightMarshal2 . . . . .	2673
6.555.3.5 tightUnmarshal . . . . .	2674
6.556activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class	
Reference . . . . .	2674
6.556.1 Detailed Description . . . . .	2675
6.556.2 Constructor & Destructor Documentation . . . . .	2675
6.556.2.1 MessageMarshaller . . . . .	2675
6.556.2.2 ~MessageMarshaller . . . . .	2675
6.556.3 Member Function Documentation . . . . .	2675
6.556.3.1 looseMarshal . . . . .	2676
6.556.3.2 looseUnmarshal . . . . .	2676
6.556.3.3 tightMarshal1 . . . . .	2677
6.556.3.4 tightMarshal2 . . . . .	2677
6.556.3.5 tightUnmarshal . . . . .	2678

6.557cms::MessageNotReadableException Class Reference . . . . .	2679
6.557.1 Detailed Description . . . . .	2679
6.557.2 Constructor & Destructor Documentation . . . . .	2679
6.557.2.1 MessageNotReadableException . . . . .	2679
6.557.2.2 MessageNotReadableException . . . . .	2679
6.557.2.3 MessageNotReadableException . . . . .	2680
6.557.2.4 MessageNotReadableException . . . . .	2680
6.557.2.5 ~MessageNotReadableException . . . . .	2680
6.558cms::MessageNotWriteableException Class Reference . . . . .	2680
6.558.1 Detailed Description . . . . .	2680
6.558.2 Constructor & Destructor Documentation . . . . .	2681
6.558.2.1 MessageNotWriteableException . . . . .	2681
6.558.2.2 MessageNotWriteableException . . . . .	2681
6.558.2.3 MessageNotWriteableException . . . . .	2681
6.558.2.4 MessageNotWriteableException . . . . .	2681
6.558.2.5 ~MessageNotWriteableException . . . . .	2681
6.559cms::MessageProducer Class Reference . . . . .	2681
6.559.1 Detailed Description . . . . .	2682
6.559.2 Constructor & Destructor Documentation . . . . .	2683
6.559.2.1 ~MessageProducer . . . . .	2683
6.559.3 Member Function Documentation . . . . .	2683
6.559.3.1 getDeliveryMode . . . . .	2683
6.559.3.2 getDisableMessageID . . . . .	2683
6.559.3.3 getDisableMessageTimeStamp . . . . .	2684
6.559.3.4 getPriority . . . . .	2684
6.559.3.5 getTimeToLive . . . . .	2684
6.559.3.6 send . . . . .	2685
6.559.3.7 send . . . . .	2685
6.559.3.8 send . . . . .	2686
6.559.3.9 send . . . . .	2687
6.559.3.10setDeliveryMode . . . . .	2687
6.559.3.11setDisableMessageID . . . . .	2688
6.559.3.12setDisableMessageTimeStamp . . . . .	2688
6.559.3.13setPriority . . . . .	2688

6.559.3.14	setTimeToLive . . . . .	2689
6.560	activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference . . . . .	2689
6.560.1	Detailed Description . . . . .	2690
6.560.2	Constructor & Destructor Documentation . . . . .	2691
6.560.2.1	MessagePropertyInterceptor . . . . .	2691
6.560.2.2	~MessagePropertyInterceptor . . . . .	2691
6.560.3	Member Function Documentation . . . . .	2691
6.560.3.1	getBooleanProperty . . . . .	2691
6.560.3.2	getByteProperty . . . . .	2691
6.560.3.3	getDoubleProperty . . . . .	2692
6.560.3.4	getFloatProperty . . . . .	2692
6.560.3.5	getIntProperty . . . . .	2692
6.560.3.6	getLongProperty . . . . .	2692
6.560.3.7	getShortProperty . . . . .	2693
6.560.3.8	getStringProperty . . . . .	2693
6.560.3.9	setBooleanProperty . . . . .	2693
6.560.3.10	setByteProperty . . . . .	2693
6.560.3.11	setDoubleProperty . . . . .	2694
6.560.3.12	setFloatProperty . . . . .	2694
6.560.3.13	setIntProperty . . . . .	2694
6.560.3.14	setLongProperty . . . . .	2694
6.560.3.15	setShortProperty . . . . .	2695
6.560.3.16	setStringProperty . . . . .	2695
6.561	activemq::commands::MessagePull Class Reference . . . . .	2695
6.561.1	Constructor & Destructor Documentation . . . . .	2696
6.561.1.1	MessagePull . . . . .	2696
6.561.1.2	~MessagePull . . . . .	2696
6.561.2	Member Function Documentation . . . . .	2696
6.561.2.1	cloneDataStructure . . . . .	2697
6.561.2.2	copyDataStructure . . . . .	2697
6.561.2.3	equals . . . . .	2697
6.561.2.4	getConsumerId . . . . .	2697
6.561.2.5	getConsumerId . . . . .	2697

6.561.2.6	getCorrelationId . . . . .	2697
6.561.2.7	getCorrelationId . . . . .	2698
6.561.2.8	getDataStructureType . . . . .	2698
6.561.2.9	getDestination . . . . .	2698
6.561.2.10	getDestination . . . . .	2698
6.561.2.11	getMessageId . . . . .	2698
6.561.2.12	getMessageId . . . . .	2698
6.561.2.13	getTimeout . . . . .	2698
6.561.2.14	setConsumerId . . . . .	2698
6.561.2.15	setCorrelationId . . . . .	2698
6.561.2.16	setDestination . . . . .	2698
6.561.2.17	setMessageId . . . . .	2698
6.561.2.18	setTimeout . . . . .	2698
6.561.2.19	toString . . . . .	2699
6.561.2.20	visit . . . . .	2699
6.561.3	Field Documentation . . . . .	2699
6.561.3.1	consumerId . . . . .	2699
6.561.3.2	correlationId . . . . .	2699
6.561.3.3	destination . . . . .	2699
6.561.3.4	ID_MESSAGEPULL . . . . .	2699
6.561.3.5	messageId . . . . .	2699
6.561.3.6	timeout . . . . .	2699
6.562	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class	
	Reference . . . . .	2700
6.562.1	Detailed Description . . . . .	2700
6.562.2	Constructor & Destructor Documentation . . . . .	2701
6.562.2.1	MessagePullMarshaller . . . . .	2701
6.562.2.2	~MessagePullMarshaller . . . . .	2701
6.562.3	Member Function Documentation . . . . .	2701
6.562.3.1	createObject . . . . .	2701
6.562.3.2	getDataStructureType . . . . .	2701
6.562.3.3	looseMarshal . . . . .	2701
6.562.3.4	looseUnmarshal . . . . .	2702
6.562.3.5	tightMarshal1 . . . . .	2702

6.562.3.6 tightMarshal2 . . . . .	2703
6.562.3.7 tightUnmarshal . . . . .	2703
6.563activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class	
Reference . . . . .	2704
6.563.1 Detailed Description . . . . .	2704
6.563.2 Constructor & Destructor Documentation . . . . .	2705
6.563.2.1 MessagePullMarshaller . . . . .	2705
6.563.2.2 ~MessagePullMarshaller . . . . .	2705
6.563.3 Member Function Documentation . . . . .	2705
6.563.3.1 createObject . . . . .	2705
6.563.3.2 getDataStructureType . . . . .	2705
6.563.3.3 looseMarshal . . . . .	2705
6.563.3.4 looseUnmarshal . . . . .	2706
6.563.3.5 tightMarshal1 . . . . .	2706
6.563.3.6 tightMarshal2 . . . . .	2707
6.563.3.7 tightUnmarshal . . . . .	2707
6.564activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class	
Reference . . . . .	2708
6.564.1 Detailed Description . . . . .	2708
6.564.2 Constructor & Destructor Documentation . . . . .	2709
6.564.2.1 MessagePullMarshaller . . . . .	2709
6.564.2.2 ~MessagePullMarshaller . . . . .	2709
6.564.3 Member Function Documentation . . . . .	2709
6.564.3.1 createObject . . . . .	2709
6.564.3.2 getDataStructureType . . . . .	2709
6.564.3.3 looseMarshal . . . . .	2709
6.564.3.4 looseUnmarshal . . . . .	2710
6.564.3.5 tightMarshal1 . . . . .	2710
6.564.3.6 tightMarshal2 . . . . .	2711
6.564.3.7 tightUnmarshal . . . . .	2711
6.565activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class	
Reference . . . . .	2712
6.565.1 Detailed Description . . . . .	2712
6.565.2 Constructor & Destructor Documentation . . . . .	2713
6.565.2.1 MessagePullMarshaller . . . . .	2713

6.565.2.2 ~MessagePullMarshaller . . . . .	2713
6.565.3 Member Function Documentation . . . . .	2713
6.565.3.1 createObject . . . . .	2713
6.565.3.2 getDataStructureType . . . . .	2713
6.565.3.3 looseMarshal . . . . .	2713
6.565.3.4 looseUnmarshal . . . . .	2714
6.565.3.5 tightMarshal1 . . . . .	2714
6.565.3.6 tightMarshal2 . . . . .	2715
6.565.3.7 tightUnmarshal . . . . .	2715
6.566activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class	
Reference . . . . .	2716
6.566.1 Detailed Description . . . . .	2716
6.566.2 Constructor & Destructor Documentation . . . . .	2717
6.566.2.1 MessagePullMarshaller . . . . .	2717
6.566.2.2 ~MessagePullMarshaller . . . . .	2717
6.566.3 Member Function Documentation . . . . .	2717
6.566.3.1 createObject . . . . .	2717
6.566.3.2 getDataStructureType . . . . .	2717
6.566.3.3 looseMarshal . . . . .	2717
6.566.3.4 looseUnmarshal . . . . .	2718
6.566.3.5 tightMarshal1 . . . . .	2718
6.566.3.6 tightMarshal2 . . . . .	2719
6.566.3.7 tightUnmarshal . . . . .	2719
6.567activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller Class	
Reference . . . . .	2720
6.567.1 Detailed Description . . . . .	2720
6.567.2 Constructor & Destructor Documentation . . . . .	2721
6.567.2.1 MessagePullMarshaller . . . . .	2721
6.567.2.2 ~MessagePullMarshaller . . . . .	2721
6.567.3 Member Function Documentation . . . . .	2721
6.567.3.1 createObject . . . . .	2721
6.567.3.2 getDataStructureType . . . . .	2721
6.567.3.3 looseMarshal . . . . .	2721
6.567.3.4 looseUnmarshal . . . . .	2722

6.567.3.5 tightMarshal1 . . . . .	2722
6.567.3.6 tightMarshal2 . . . . .	2723
6.567.3.7 tightUnmarshal . . . . .	2723
6.568activemq::transport::mock::MockTransport Class Reference . . . . .	2724
6.568.1 Detailed Description . . . . .	2726
6.568.2 Constructor & Destructor Documentation . . . . .	2726
6.568.2.1 MockTransport . . . . .	2726
6.568.2.2 ~MockTransport . . . . .	2726
6.568.3 Member Function Documentation . . . . .	2726
6.568.3.1 close . . . . .	2726
6.568.3.2 fireCommand . . . . .	2726
6.568.3.3 fireException . . . . .	2727
6.568.3.4 getInstance . . . . .	2727
6.568.3.5 getNumReceivedMessageBeforeFail . . . . .	2727
6.568.3.6 getNumReceivedMessages . . . . .	2727
6.568.3.7 getNumSentKeepAlives . . . . .	2727
6.568.3.8 getNumSentKeepAlivesBeforeFail . . . . .	2727
6.568.3.9 getNumSentMessageBeforeFail . . . . .	2727
6.568.3.10getNumSentMessages . . . . .	2727
6.568.3.11getRemoteAddress . . . . .	2727
6.568.3.12getTransportListener . . . . .	2728
6.568.3.13getWireFormat . . . . .	2728
6.568.3.14sClosed . . . . .	2728
6.568.3.15sConnected . . . . .	2728
6.568.3.16sFailOnClose . . . . .	2729
6.568.3.17sFailOnKeepAliveSends . . . . .	2729
6.568.3.18sFailOnReceiveMessage . . . . .	2729
6.568.3.19sFailOnSendMessage . . . . .	2729
6.568.3.20sFailOnStart . . . . .	2729
6.568.3.21isFailOnStop . . . . .	2729
6.568.3.22sFaultTolerant . . . . .	2729
6.568.3.23narrow . . . . .	2729
6.568.3.24oneway . . . . .	2730
6.568.3.25reconnect . . . . .	2730



6.568.3.26request . . . . .	2730
6.568.3.27request . . . . .	2731
6.568.3.28setFailOnClose . . . . .	2731
6.568.3.29setFailOnKeepAliveSends . . . . .	2731
6.568.3.30setFailOnReceiveMessage . . . . .	2731
6.568.3.31setFailOnSendMessage . . . . .	2731
6.568.3.32setFailOnStart . . . . .	2731
6.568.3.33setFailOnStop . . . . .	2731
6.568.3.34setNumReceivedMessageBeforeFail . . . . .	2732
6.568.3.35setNumReceivedMessages . . . . .	2732
6.568.3.36setNumSentKeepAlives . . . . .	2732
6.568.3.37setNumSentKeepAlivesBeforeFail . . . . .	2732
6.568.3.38setNumSentMessageBeforeFail . . . . .	2732
6.568.3.39setNumSentMessages . . . . .	2732
6.568.3.40setOutgoingListener . . . . .	2732
6.568.3.41setResponseBuilder . . . . .	2732
6.568.3.42setTransportListener . . . . .	2732
6.568.3.43setWireFormat . . . . .	2733
6.568.3.44start . . . . .	2733
6.568.3.45stop . . . . .	2733
6.569activemq::transport::mock::MockTransportFactory Class Reference . . .	2734
6.569.1 Detailed Description . . . . .	2734
6.569.2 Constructor & Destructor Documentation . . . . .	2734
6.569.2.1 ~MockTransportFactory . . . . .	2734
6.569.3 Member Function Documentation . . . . .	2734
6.569.3.1 create . . . . .	2735
6.569.3.2 createComposite . . . . .	2735
6.569.3.3 doCreateComposite . . . . .	2735
6.570decaf::util::concurrent::Mutex Class Reference . . . . .	2736
6.570.1 Detailed Description . . . . .	2737
6.570.2 Constructor & Destructor Documentation . . . . .	2737
6.570.2.1 Mutex . . . . .	2737
6.570.2.2 ~Mutex . . . . .	2737
6.570.3 Member Function Documentation . . . . .	2737

6.570.3.1 lock . . . . .	2737
6.570.3.2 notify . . . . .	2737
6.570.3.3 notifyAll . . . . .	2738
6.570.3.4 tryLock . . . . .	2738
6.570.3.5 unlock . . . . .	2739
6.570.3.6 wait . . . . .	2739
6.570.3.7 wait . . . . .	2740
6.570.3.8 wait . . . . .	2740
6.571decaf::util::concurrent::MutexHandle Class Reference . . . . .	2741
6.571.1 Constructor & Destructor Documentation . . . . .	2741
6.571.1.1 MutexHandle . . . . .	2741
6.571.1.2 ~MutexHandle . . . . .	2741
6.571.1.3 MutexHandle . . . . .	2741
6.571.1.4 ~MutexHandle . . . . .	2741
6.571.2 Field Documentation . . . . .	2741
6.571.2.1 lock_count . . . . .	2741
6.571.2.2 lock_owner . . . . .	2741
6.571.2.3 mutex . . . . .	2742
6.571.2.4 mutex . . . . .	2742
6.572decaf::internal::util::concurrent::MutexImpl Class Reference . . . . .	2742
6.572.1 Member Function Documentation . . . . .	2742
6.572.1.1 create . . . . .	2742
6.572.1.2 destroy . . . . .	2742
6.572.1.3 lock . . . . .	2743
6.572.1.4 trylock . . . . .	2743
6.572.1.5 unlock . . . . .	2743
6.573decaf::internal::net::Network Class Reference . . . . .	2744
6.573.1 Detailed Description . . . . .	2744
6.573.2 Constructor & Destructor Documentation . . . . .	2745
6.573.2.1 Network . . . . .	2745
6.573.2.2 ~Network . . . . .	2745
6.573.3 Member Function Documentation . . . . .	2745
6.573.3.1 addAsResource . . . . .	2745
6.573.3.2 addNetworkResource . . . . .	2745

6.573.3.3	getNetworkRuntime	2745
6.573.3.4	getRuntimeLock	2745
6.573.3.5	initializeNetworking	2746
6.573.3.6	shutdownNetworking	2746
6.574	activemq::commands::NetworkBridgeFilter Class Reference	2746
6.574.1	Constructor & Destructor Documentation	2747
6.574.1.1	NetworkBridgeFilter	2747
6.574.1.2	~NetworkBridgeFilter	2747
6.574.2	Member Function Documentation	2747
6.574.2.1	cloneDataStructure	2747
6.574.2.2	copyDataStructure	2747
6.574.2.3	equals	2748
6.574.2.4	getDataStructureType	2748
6.574.2.5	getNetworkBrokerId	2748
6.574.2.6	getNetworkBrokerId	2748
6.574.2.7	getNetworkTTL	2748
6.574.2.8	setNetworkBrokerId	2748
6.574.2.9	setNetworkTTL	2748
6.574.2.10	toString	2748
6.574.3	Field Documentation	2749
6.574.3.1	ID_NETWORKBRIDGEFILTER	2749
6.574.3.2	networkBrokerId	2749
6.574.3.3	networkTTL	2749
6.575	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference	2749
6.575.1	Detailed Description	2750
6.575.2	Constructor & Destructor Documentation	2750
6.575.2.1	NetworkBridgeFilterMarshaller	2750
6.575.2.2	~NetworkBridgeFilterMarshaller	2750
6.575.3	Member Function Documentation	2750
6.575.3.1	createObject	2750
6.575.3.2	getDataStructureType	2751
6.575.3.3	looseMarshal	2751
6.575.3.4	looseUnmarshal	2751

6.575.3.5 tightMarshal1 . . . . .	2752
6.575.3.6 tightMarshal2 . . . . .	2752
6.575.3.7 tightUnmarshal . . . . .	2753
6.576activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2753
6.576.1 Detailed Description . . . . .	2754
6.576.2 Constructor & Destructor Documentation . . . . .	2754
6.576.2.1 NetworkBridgeFilterMarshaller . . . . .	2754
6.576.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2754
6.576.3 Member Function Documentation . . . . .	2754
6.576.3.1 createObject . . . . .	2754
6.576.3.2 getDataStructureType . . . . .	2755
6.576.3.3 looseMarshal . . . . .	2755
6.576.3.4 looseUnmarshal . . . . .	2755
6.576.3.5 tightMarshal1 . . . . .	2756
6.576.3.6 tightMarshal2 . . . . .	2756
6.576.3.7 tightUnmarshal . . . . .	2757
6.577activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2757
6.577.1 Detailed Description . . . . .	2758
6.577.2 Constructor & Destructor Documentation . . . . .	2758
6.577.2.1 NetworkBridgeFilterMarshaller . . . . .	2758
6.577.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2758
6.577.3 Member Function Documentation . . . . .	2758
6.577.3.1 createObject . . . . .	2758
6.577.3.2 getDataStructureType . . . . .	2759
6.577.3.3 looseMarshal . . . . .	2759
6.577.3.4 looseUnmarshal . . . . .	2759
6.577.3.5 tightMarshal1 . . . . .	2760
6.577.3.6 tightMarshal2 . . . . .	2760
6.577.3.7 tightUnmarshal . . . . .	2761
6.578activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2761
6.578.1 Detailed Description . . . . .	2762
6.578.2 Constructor & Destructor Documentation . . . . .	2762

6.578.2.1 NetworkBridgeFilterMarshaller . . . . .	2762
6.578.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2762
6.578.3 Member Function Documentation . . . . .	2762
6.578.3.1 createObject . . . . .	2762
6.578.3.2 getDataStructureType . . . . .	2763
6.578.3.3 looseMarshal . . . . .	2763
6.578.3.4 looseUnmarshal . . . . .	2763
6.578.3.5 tightMarshal1 . . . . .	2764
6.578.3.6 tightMarshal2 . . . . .	2764
6.578.3.7 tightUnmarshal . . . . .	2765
6.579activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2765
6.579.1 Detailed Description . . . . .	2766
6.579.2 Constructor & Destructor Documentation . . . . .	2766
6.579.2.1 NetworkBridgeFilterMarshaller . . . . .	2766
6.579.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2766
6.579.3 Member Function Documentation . . . . .	2766
6.579.3.1 createObject . . . . .	2766
6.579.3.2 getDataStructureType . . . . .	2767
6.579.3.3 looseMarshal . . . . .	2767
6.579.3.4 looseUnmarshal . . . . .	2767
6.579.3.5 tightMarshal1 . . . . .	2768
6.579.3.6 tightMarshal2 . . . . .	2768
6.579.3.7 tightUnmarshal . . . . .	2769
6.580activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	
Class Reference . . . . .	2769
6.580.1 Detailed Description . . . . .	2770
6.580.2 Constructor & Destructor Documentation . . . . .	2770
6.580.2.1 NetworkBridgeFilterMarshaller . . . . .	2770
6.580.2.2 ~NetworkBridgeFilterMarshaller . . . . .	2770
6.580.3 Member Function Documentation . . . . .	2770
6.580.3.1 createObject . . . . .	2770
6.580.3.2 getDataStructureType . . . . .	2771
6.580.3.3 looseMarshal . . . . .	2771

6.580.3.4 looseUnmarshal . . . . .	2771
6.580.3.5 tightMarshal1 . . . . .	2772
6.580.3.6 tightMarshal2 . . . . .	2772
6.580.3.7 tightUnmarshal . . . . .	2773
6.581decaf::net::NoRouteToHostException Class Reference . . . . .	2773
6.581.1 Constructor & Destructor Documentation . . . . .	2774
6.581.1.1 NoRouteToHostException . . . . .	2774
6.581.1.2 NoRouteToHostException . . . . .	2774
6.581.1.3 NoRouteToHostException . . . . .	2774
6.581.1.4 NoRouteToHostException . . . . .	2774
6.581.1.5 NoRouteToHostException . . . . .	2775
6.581.1.6 NoRouteToHostException . . . . .	2775
6.581.1.7 ~NoRouteToHostException . . . . .	2775
6.581.2 Member Function Documentation . . . . .	2775
6.581.2.1 clone . . . . .	2775
6.582decaf::security::NoSuchAlgorithmException Class Reference . . . . .	2776
6.582.1 Constructor & Destructor Documentation . . . . .	2776
6.582.1.1 NoSuchAlgorithmException . . . . .	2776
6.582.1.2 NoSuchAlgorithmException . . . . .	2777
6.582.1.3 NoSuchAlgorithmException . . . . .	2777
6.582.1.4 NoSuchAlgorithmException . . . . .	2777
6.582.1.5 NoSuchAlgorithmException . . . . .	2777
6.582.1.6 NoSuchAlgorithmException . . . . .	2778
6.582.1.7 ~NoSuchAlgorithmException . . . . .	2778
6.582.2 Member Function Documentation . . . . .	2778
6.582.2.1 clone . . . . .	2778
6.583decaf::lang::exceptions::NoSuchElementException Class Reference . . . . .	2778
6.583.1 Constructor & Destructor Documentation . . . . .	2779
6.583.1.1 NoSuchElementException . . . . .	2779
6.583.1.2 NoSuchElementException . . . . .	2779
6.583.1.3 NoSuchElementException . . . . .	2779
6.583.1.4 NoSuchElementException . . . . .	2780
6.583.1.5 NoSuchElementException . . . . .	2780
6.583.1.6 NoSuchElementException . . . . .	2780

6.583.1.7 ~NoSuchElementException . . . . .	2780
6.583.2 Member Function Documentation . . . . .	2780
6.583.2.1 clone . . . . .	2781
6.584decaf::security::NoSuchProviderException Class Reference . . . . .	2781
6.584.1 Constructor & Destructor Documentation . . . . .	2782
6.584.1.1 NoSuchProviderException . . . . .	2782
6.584.1.2 NoSuchProviderException . . . . .	2782
6.584.1.3 NoSuchProviderException . . . . .	2782
6.584.1.4 NoSuchProviderException . . . . .	2782
6.584.1.5 NoSuchProviderException . . . . .	2782
6.584.1.6 NoSuchProviderException . . . . .	2783
6.584.1.7 ~NoSuchProviderException . . . . .	2783
6.584.2 Member Function Documentation . . . . .	2783
6.584.2.1 clone . . . . .	2783
6.585decaf::lang::exceptions::NullPointerException Class Reference . . . . .	2783
6.585.1 Constructor & Destructor Documentation . . . . .	2784
6.585.1.1 NullPointerException . . . . .	2784
6.585.1.2 NullPointerException . . . . .	2784
6.585.1.3 NullPointerException . . . . .	2784
6.585.1.4 NullPointerException . . . . .	2785
6.585.1.5 NullPointerException . . . . .	2785
6.585.1.6 NullPointerException . . . . .	2785
6.585.1.7 ~NullPointerException . . . . .	2786
6.585.2 Member Function Documentation . . . . .	2786
6.585.2.1 clone . . . . .	2786
6.586decaf::lang::Number Class Reference . . . . .	2786
6.586.1 Detailed Description . . . . .	2787
6.586.2 Constructor & Destructor Documentation . . . . .	2787
6.586.2.1 ~Number . . . . .	2787
6.586.3 Member Function Documentation . . . . .	2787
6.586.3.1 byteValue . . . . .	2787
6.586.3.2 doubleValue . . . . .	2787
6.586.3.3 floatValue . . . . .	2787
6.586.3.4 intValue . . . . .	2788

6.586.3.5 longValue . . . . .	2788
6.586.3.6 shortValue . . . . .	2788
6.587decaf::lang::exceptions::NumberFormatException Class Reference . . .	2789
6.587.1 Constructor & Destructor Documentation . . . . .	2789
6.587.1.1 NumberFormatException . . . . .	2789
6.587.1.2 NumberFormatException . . . . .	2789
6.587.1.3 NumberFormatException . . . . .	2790
6.587.1.4 NumberFormatException . . . . .	2790
6.587.1.5 NumberFormatException . . . . .	2790
6.587.1.6 NumberFormatException . . . . .	2790
6.587.1.7 ~NumberFormatException . . . . .	2791
6.587.2 Member Function Documentation . . . . .	2791
6.587.2.1 clone . . . . .	2791
6.588cms::ObjectMessage Class Reference . . . . .	2791
6.588.1 Detailed Description . . . . .	2792
6.588.2 Constructor & Destructor Documentation . . . . .	2792
6.588.2.1 ~ObjectMessage . . . . .	2792
6.589decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference .	2792
6.589.1 Detailed Description . . . . .	2793
6.589.2 Constructor & Destructor Documentation . . . . .	2793
6.589.2.1 OpenSSLContextSpi . . . . .	2793
6.589.2.2 ~OpenSSLContextSpi . . . . .	2793
6.589.3 Member Function Documentation . . . . .	2793
6.589.3.1 providerGetServerSocketFactory . . . . .	2794
6.589.3.2 providerGetSocketFactory . . . . .	2794
6.589.3.3 providerInit . . . . .	2794
6.589.4 Friends And Related Function Documentation . . . . .	2795
6.589.4.1 OpenSSLSocket . . . . .	2795
6.589.4.2 OpenSSLSocketFactory . . . . .	2795
6.590decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference	2795
6.590.1 Detailed Description . . . . .	2796
6.590.2 Constructor & Destructor Documentation . . . . .	2796
6.590.2.1 ~OpenSSLParameters . . . . .	2796
6.590.3 Member Function Documentation . . . . .	2796



6.590.3.1 clone . . . . .	2796
6.590.3.2 getEnabledCipherSuites . . . . .	2796
6.590.3.3 getEnabledProtocols . . . . .	2796
6.590.3.4 getNeedClientAuth . . . . .	2796
6.590.3.5 getSupportedCipherSuites . . . . .	2796
6.590.3.6 getSupportedProtocols . . . . .	2796
6.590.3.7 getUseClientMode . . . . .	2796
6.590.3.8 getWantClientAuth . . . . .	2796
6.590.3.9 setEnabledCipherSuites . . . . .	2796
6.590.3.10setEnabledProtocols . . . . .	2796
6.590.3.11setNeedClientAuth . . . . .	2797
6.590.3.12setUseClientMode . . . . .	2797
6.590.3.13setWantClientAuth . . . . .	2797
6.591decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference	2797
6.591.1 Detailed Description . . . . .	2799
6.591.2 Constructor & Destructor Documentation . . . . .	2799
6.591.2.1 OpenSSLServerSocket . . . . .	2799
6.591.2.2 ~OpenSSLServerSocket . . . . .	2799
6.591.3 Member Function Documentation . . . . .	2799
6.591.3.1 accept . . . . .	2799
6.591.3.2 getEnabledCipherSuites . . . . .	2800
6.591.3.3 getEnabledProtocols . . . . .	2800
6.591.3.4 getNeedClientAuth . . . . .	2800
6.591.3.5 getSupportedCipherSuites . . . . .	2801
6.591.3.6 getSupportedProtocols . . . . .	2801
6.591.3.7 getWantClientAuth . . . . .	2801
6.591.3.8 setEnabledCipherSuites . . . . .	2801
6.591.3.9 setEnabledProtocols . . . . .	2802
6.591.3.10setNeedClientAuth . . . . .	2802
6.591.3.11setWantClientAuth . . . . .	2802
6.592decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory Class Reference . . . . .	2803
6.592.1 Detailed Description . . . . .	2805
6.592.2 Constructor & Destructor Documentation . . . . .	2805

6.592.2.1	OpenSSLServerSocketFactory . . . . .	2805
6.592.2.2	~OpenSSLServerSocketFactory . . . . .	2805
6.592.3	Member Function Documentation . . . . .	2805
6.592.3.1	createServerSocket . . . . .	2805
6.592.3.2	createServerSocket . . . . .	2805
6.592.3.3	createServerSocket . . . . .	2806
6.592.3.4	createServerSocket . . . . .	2806
6.592.3.5	getDefaultCipherSuites . . . . .	2807
6.592.3.6	getSupportedCipherSuites . . . . .	2807
6.593	decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference . . .	2808
6.593.1	Detailed Description . . . . .	2812
6.593.2	Constructor & Destructor Documentation . . . . .	2812
6.593.2.1	OpenSSLSocket . . . . .	2812
6.593.2.2	OpenSSLSocket . . . . .	2812
6.593.2.3	OpenSSLSocket . . . . .	2812
6.593.2.4	OpenSSLSocket . . . . .	2813
6.593.2.5	OpenSSLSocket . . . . .	2813
6.593.2.6	~OpenSSLSocket . . . . .	2813
6.593.3	Member Function Documentation . . . . .	2813
6.593.3.1	available . . . . .	2813
6.593.3.2	close . . . . .	2813
6.593.3.3	connect . . . . .	2813
6.593.3.4	getEnabledCipherSuites . . . . .	2814
6.593.3.5	getEnabledProtocols . . . . .	2814
6.593.3.6	getInputStream . . . . .	2814
6.593.3.7	getNeedClientAuth . . . . .	2815
6.593.3.8	getOutputStream . . . . .	2815
6.593.3.9	getSupportedCipherSuites . . . . .	2816
6.593.3.10	getSupportedProtocols . . . . .	2816
6.593.3.11	getUseClientMode . . . . .	2816
6.593.3.12	getWantClientAuth . . . . .	2817
6.593.3.13	read . . . . .	2817
6.593.3.14	sendUrgentData . . . . .	2817
6.593.3.15	setEnabledCipherSuites . . . . .	2818

6.593.3.16	setEnabledProtocols . . . . .	2818
6.593.3.17	setNeedClientAuth . . . . .	2818
6.593.3.18	setOOBInline . . . . .	2819
6.593.3.19	setUseClientMode . . . . .	2819
6.593.3.20	setWantClientAuth . . . . .	2820
6.593.3.21	shutdownInput . . . . .	2820
6.593.3.22	shutdownOutput . . . . .	2820
6.593.3.23	startHandshake . . . . .	2820
6.593.3.24	write . . . . .	2821
6.594	decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference . . . . .	2821
6.594.1	Detailed Description . . . . .	2822
6.594.2	Constructor & Destructor Documentation . . . . .	2822
6.594.2.1	OpenSSLSocketException . . . . .	2822
6.594.2.2	OpenSSLSocketException . . . . .	2823
6.594.2.3	OpenSSLSocketException . . . . .	2823
6.594.2.4	OpenSSLSocketException . . . . .	2823
6.594.2.5	OpenSSLSocketException . . . . .	2823
6.594.2.6	OpenSSLSocketException . . . . .	2824
6.594.2.7	OpenSSLSocketException . . . . .	2824
6.594.2.8	~OpenSSLSocketException . . . . .	2824
6.594.3	Member Function Documentation . . . . .	2824
6.594.3.1	clone . . . . .	2824
6.594.3.2	getErrorString . . . . .	2825
6.595	decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference2825	
6.595.1	Detailed Description . . . . .	2828
6.595.2	Constructor & Destructor Documentation . . . . .	2828
6.595.2.1	OpenSSLSocketFactory . . . . .	2828
6.595.2.2	~OpenSSLSocketFactory . . . . .	2828
6.595.3	Member Function Documentation . . . . .	2828
6.595.3.1	createSocket . . . . .	2828
6.595.3.2	createSocket . . . . .	2828
6.595.3.3	createSocket . . . . .	2829
6.595.3.4	createSocket . . . . .	2830

6.595.3.5 createSocket . . . . .	2830
6.595.3.6 createSocket . . . . .	2831
6.595.3.7 getDefaultCipherSuites . . . . .	2831
6.595.3.8 getSupportedCipherSuites . . . . .	2832
6.596decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream Class Reference . . . . .	2832
6.596.1 Detailed Description . . . . .	2833
6.596.2 Constructor & Destructor Documentation . . . . .	2833
6.596.2.1 OpenSSLSocketInputStream . . . . .	2833
6.596.2.2 ~OpenSSLSocketInputStream . . . . .	2833
6.596.3 Member Function Documentation . . . . .	2833
6.596.3.1 available . . . . .	2833
6.596.3.2 close . . . . .	2834
6.596.3.3 doReadArrayBounded . . . . .	2834
6.596.3.4 doReadByte . . . . .	2834
6.596.3.5 skip . . . . .	2834
6.597decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream Class Reference . . . . .	2835
6.597.1 Detailed Description . . . . .	2836
6.597.2 Constructor & Destructor Documentation . . . . .	2836
6.597.2.1 OpenSSLSocketOutputStream . . . . .	2836
6.597.2.2 ~OpenSSLSocketOutputStream . . . . .	2836
6.597.3 Member Function Documentation . . . . .	2836
6.597.3.1 close . . . . .	2836
6.597.3.2 doWriteArrayBounded . . . . .	2836
6.597.3.3 doWriteByte . . . . .	2837
6.598activemq::wireformat::openwire::OpenWireFormat Class Reference . . . . .	2837
6.598.1 Constructor & Destructor Documentation . . . . .	2839
6.598.1.1 OpenWireFormat . . . . .	2839
6.598.1.2 ~OpenWireFormat . . . . .	2840
6.598.2 Member Function Documentation . . . . .	2840
6.598.2.1 addMarshaller . . . . .	2840
6.598.2.2 createNegotiator . . . . .	2840
6.598.2.3 destroyMarshalers . . . . .	2840

6.598.2.4 doUnmarshal . . . . .	2841
6.598.2.5 getCacheSize . . . . .	2841
6.598.2.6 getMaxInactivityDuration . . . . .	2841
6.598.2.7 getMaxInactivityDurationInitialDelay . . . . .	2841
6.598.2.8 getPreferredWireFormatInfo . . . . .	2842
6.598.2.9 getVersion . . . . .	2842
6.598.2.10hasNegotiator . . . . .	2842
6.598.2.11inReceive . . . . .	2842
6.598.2.12sCacheEnabled . . . . .	2843
6.598.2.13sSizePrefixDisabled . . . . .	2843
6.598.2.14sStackTraceEnabled . . . . .	2843
6.598.2.15sTcpNoDelayEnabled . . . . .	2843
6.598.2.16sTightEncodingEnabled . . . . .	2843
6.598.2.17ooseMarshalNestedObject . . . . .	2844
6.598.2.18ooseUnmarshalNestedObject . . . . .	2844
6.598.2.19marshal . . . . .	2844
6.598.2.20renegotiateWireFormat . . . . .	2845
6.598.2.21setCacheEnabled . . . . .	2845
6.598.2.22setCacheSize . . . . .	2845
6.598.2.23setMaxInactivityDuration . . . . .	2846
6.598.2.24setMaxInactivityDurationInitialDelay . . . . .	2846
6.598.2.25setPreferredWireFormatInfo . . . . .	2846
6.598.2.26setSizePrefixDisabled . . . . .	2846
6.598.2.27setStackTraceEnabled . . . . .	2846
6.598.2.28setTcpNoDelayEnabled . . . . .	2847
6.598.2.29setTightEncodingEnabled . . . . .	2847
6.598.2.30setVersion . . . . .	2847
6.598.2.31tightMarshalNestedObject1 . . . . .	2847
6.598.2.32tightMarshalNestedObject2 . . . . .	2848
6.598.2.33tightUnmarshalNestedObject . . . . .	2848
6.598.2.34unmarshal . . . . .	2849
6.598.3 Field Documentation . . . . .	2849
6.598.3.1 DEFAULT_VERSION . . . . .	2849
6.598.3.2 NULL_TYPE . . . . .	2849

6.599activemq::wireformat::openwire::OpenWireFormatFactory Class Reference . . . . .	2849
6.599.1 Constructor & Destructor Documentation . . . . .	2850
6.599.1.1 OpenWireFormatFactory . . . . .	2850
6.599.1.2 ~OpenWireFormatFactory . . . . .	2850
6.599.2 Member Function Documentation . . . . .	2850
6.599.2.1 createWireFormat . . . . .	2850
6.600activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference . . . . .	2851
6.600.1 Constructor & Destructor Documentation . . . . .	2851
6.600.1.1 OpenWireFormatNegotiator . . . . .	2851
6.600.1.2 ~OpenWireFormatNegotiator . . . . .	2852
6.600.2 Member Function Documentation . . . . .	2852
6.600.2.1 close . . . . .	2852
6.600.2.2 onCommand . . . . .	2852
6.600.2.3 oneway . . . . .	2852
6.600.2.4 onTransportException . . . . .	2853
6.600.2.5 request . . . . .	2853
6.600.2.6 request . . . . .	2854
6.600.2.7 start . . . . .	2854
6.601activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference . . . . .	2854
6.601.1 Constructor & Destructor Documentation . . . . .	2855
6.601.1.1 OpenWireResponseBuilder . . . . .	2855
6.601.1.2 ~OpenWireResponseBuilder . . . . .	2855
6.601.2 Member Function Documentation . . . . .	2855
6.601.2.1 buildIncomingCommands . . . . .	2855
6.601.2.2 buildResponse . . . . .	2856
6.602decaf::io::OutputStream Class Reference . . . . .	2856
6.602.1 Detailed Description . . . . .	2858
6.602.2 Constructor & Destructor Documentation . . . . .	2858
6.602.2.1 OutputStream . . . . .	2858
6.602.2.2 ~OutputStream . . . . .	2858
6.602.3 Member Function Documentation . . . . .	2858
6.602.3.1 close . . . . .	2858

6.602.3.2 doWriteArray . . . . .	2858
6.602.3.3 doWriteArrayBounded . . . . .	2859
6.602.3.4 doWriteByte . . . . .	2859
6.602.3.5 flush . . . . .	2859
6.602.3.6 lock . . . . .	2859
6.602.3.7 notify . . . . .	2860
6.602.3.8 notifyAll . . . . .	2860
6.602.3.9 toString . . . . .	2860
6.602.3.10 tryLock . . . . .	2861
6.602.3.11 unlock . . . . .	2861
6.602.3.12 wait . . . . .	2861
6.602.3.13 wait . . . . .	2862
6.602.3.14 wait . . . . .	2862
6.602.3.15 write . . . . .	2863
6.602.3.16 write . . . . .	2863
6.602.3.17 write . . . . .	2864
6.603 decaf::io::OutputStreamWriter Class Reference . . . . .	2864
6.603.1 Detailed Description . . . . .	2865
6.603.2 Constructor & Destructor Documentation . . . . .	2865
6.603.2.1 OutputStreamWriter . . . . .	2865
6.603.2.2 ~OutputStreamWriter . . . . .	2865
6.603.3 Member Function Documentation . . . . .	2865
6.603.3.1 checkClosed . . . . .	2866
6.603.3.2 close . . . . .	2866
6.603.3.3 doWriteArrayBounded . . . . .	2866
6.603.3.4 flush . . . . .	2866
6.604 activemq::commands::PartialCommand Class Reference . . . . .	2866
6.604.1 Constructor & Destructor Documentation . . . . .	2867
6.604.1.1 PartialCommand . . . . .	2867
6.604.1.2 ~PartialCommand . . . . .	2867
6.604.2 Member Function Documentation . . . . .	2868
6.604.2.1 cloneDataStructure . . . . .	2868
6.604.2.2 copyDataStructure . . . . .	2868
6.604.2.3 equals . . . . .	2868

6.604.2.4	getCommandId . . . . .	2868
6.604.2.5	getData . . . . .	2869
6.604.2.6	getData . . . . .	2869
6.604.2.7	getDataStructureType . . . . .	2869
6.604.2.8	setCommandId . . . . .	2869
6.604.2.9	setData . . . . .	2869
6.604.2.10	toString . . . . .	2869
6.604.3	Field Documentation . . . . .	2869
6.604.3.1	commandId . . . . .	2869
6.604.3.2	data . . . . .	2869
6.604.3.3	ID_PARTIALCOMMAND . . . . .	2870
6.605	activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller	
	Class Reference . . . . .	2870
6.605.1	Detailed Description . . . . .	2871
6.605.2	Constructor & Destructor Documentation . . . . .	2871
6.605.2.1	PartialCommandMarshaller . . . . .	2871
6.605.2.2	~PartialCommandMarshaller . . . . .	2871
6.605.3	Member Function Documentation . . . . .	2871
6.605.3.1	createObject . . . . .	2871
6.605.3.2	getDataStructureType . . . . .	2871
6.605.3.3	looseMarshal . . . . .	2872
6.605.3.4	looseUnmarshal . . . . .	2872
6.605.3.5	tightMarshal1 . . . . .	2873
6.605.3.6	tightMarshal2 . . . . .	2873
6.605.3.7	tightUnmarshal . . . . .	2874
6.606	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	
	Class Reference . . . . .	2874
6.606.1	Detailed Description . . . . .	2875
6.606.2	Constructor & Destructor Documentation . . . . .	2875
6.606.2.1	PartialCommandMarshaller . . . . .	2875
6.606.2.2	~PartialCommandMarshaller . . . . .	2875
6.606.3	Member Function Documentation . . . . .	2875
6.606.3.1	createObject . . . . .	2875
6.606.3.2	getDataStructureType . . . . .	2876



6.606.3.3 looseMarshal . . . . .	2876
6.606.3.4 looseUnmarshal . . . . .	2876
6.606.3.5 tightMarshal1 . . . . .	2877
6.606.3.6 tightMarshal2 . . . . .	2877
6.606.3.7 tightUnmarshal . . . . .	2878
6.607activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	
Class Reference . . . . .	2878
6.607.1 Detailed Description . . . . .	2879
6.607.2 Constructor & Destructor Documentation . . . . .	2879
6.607.2.1 PartialCommandMarshaller . . . . .	2879
6.607.2.2 ~PartialCommandMarshaller . . . . .	2880
6.607.3 Member Function Documentation . . . . .	2880
6.607.3.1 createObject . . . . .	2880
6.607.3.2 getDataStructureType . . . . .	2880
6.607.3.3 looseMarshal . . . . .	2880
6.607.3.4 looseUnmarshal . . . . .	2881
6.607.3.5 tightMarshal1 . . . . .	2881
6.607.3.6 tightMarshal2 . . . . .	2882
6.607.3.7 tightUnmarshal . . . . .	2882
6.608activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	
Class Reference . . . . .	2883
6.608.1 Detailed Description . . . . .	2884
6.608.2 Constructor & Destructor Documentation . . . . .	2884
6.608.2.1 PartialCommandMarshaller . . . . .	2884
6.608.2.2 ~PartialCommandMarshaller . . . . .	2884
6.608.3 Member Function Documentation . . . . .	2884
6.608.3.1 createObject . . . . .	2884
6.608.3.2 getDataStructureType . . . . .	2884
6.608.3.3 looseMarshal . . . . .	2885
6.608.3.4 looseUnmarshal . . . . .	2885
6.608.3.5 tightMarshal1 . . . . .	2886
6.608.3.6 tightMarshal2 . . . . .	2886
6.608.3.7 tightUnmarshal . . . . .	2887
6.609activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	
Class Reference . . . . .	2887

6.609.1 Detailed Description . . . . .	2888
6.609.2 Constructor & Destructor Documentation . . . . .	2888
6.609.2.1 PartialCommandMarshaller . . . . .	2888
6.609.2.2 ~PartialCommandMarshaller . . . . .	2888
6.609.3 Member Function Documentation . . . . .	2888
6.609.3.1 createObject . . . . .	2888
6.609.3.2 getDataStructureType . . . . .	2889
6.609.3.3 looseMarshal . . . . .	2889
6.609.3.4 looseUnmarshal . . . . .	2889
6.609.3.5 tightMarshal1 . . . . .	2890
6.609.3.6 tightMarshal2 . . . . .	2890
6.609.3.7 tightUnmarshal . . . . .	2891
6.610activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	
Class Reference . . . . .	2891
6.610.1 Detailed Description . . . . .	2892
6.610.2 Constructor & Destructor Documentation . . . . .	2892
6.610.2.1 PartialCommandMarshaller . . . . .	2892
6.610.2.2 ~PartialCommandMarshaller . . . . .	2893
6.610.3 Member Function Documentation . . . . .	2893
6.610.3.1 createObject . . . . .	2893
6.610.3.2 getDataStructureType . . . . .	2893
6.610.3.3 looseMarshal . . . . .	2893
6.610.3.4 looseUnmarshal . . . . .	2894
6.610.3.5 tightMarshal1 . . . . .	2894
6.610.3.6 tightMarshal2 . . . . .	2895
6.610.3.7 tightUnmarshal . . . . .	2895
6.611decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference . . . . .	2896
6.611.1 Detailed Description . . . . .	2897
6.611.2 Member Typedef Documentation . . . . .	2898
6.611.2.1 CounterType . . . . .	2898
6.611.2.2 PointerType . . . . .	2898
6.611.2.3 ReferenceType . . . . .	2898
6.611.3 Constructor & Destructor Documentation . . . . .	2898
6.611.3.1 Pointer . . . . .	2898

6.611.3.2 Pointer . . . . .	2898
6.611.3.3 Pointer . . . . .	2899
6.611.3.4 Pointer . . . . .	2899
6.611.3.5 Pointer . . . . .	2899
6.611.3.6 Pointer . . . . .	2899
6.611.3.7 ~Pointer . . . . .	2900
6.611.4 Member Function Documentation . . . . .	2900
6.611.4.1 dynamicCast . . . . .	2900
6.611.4.2 get . . . . .	2900
6.611.4.3 operator! . . . . .	2900
6.611.4.4 operator!= . . . . .	2900
6.611.4.5 operator* . . . . .	2901
6.611.4.6 operator* . . . . .	2901
6.611.4.7 operator-> . . . . .	2901
6.611.4.8 operator-> . . . . .	2901
6.611.4.9 operator= . . . . .	2901
6.611.4.10operator= . . . . .	2902
6.611.4.11operator== . . . . .	2902
6.611.4.12release . . . . .	2902
6.611.4.13reset . . . . .	2902
6.611.4.14staticCast . . . . .	2902
6.611.4.15swap . . . . .	2903
6.611.5 Friends And Related Function Documentation . . . . .	2903
6.611.5.1 operator!= . . . . .	2903
6.611.5.2 operator!= . . . . .	2903
6.611.5.3 operator== . . . . .	2903
6.611.5.4 operator== . . . . .	2903
6.612decaf::lang::PointerComparator< T, R > Class Template Reference . . .	2903
6.612.1 Detailed Description . . . . .	2904
6.612.2 Member Function Documentation . . . . .	2904
6.612.2.1 compare . . . . .	2904
6.612.2.2 operator() . . . . .	2904
6.613activemq::cmsutil::PooledSession Class Reference . . . . .	2904
6.613.1 Detailed Description . . . . .	2907

6.613.2 Constructor & Destructor Documentation . . . . .	2907
6.613.2.1 PooledSession . . . . .	2907
6.613.2.2 PooledSession . . . . .	2907
6.613.2.3 ~PooledSession . . . . .	2907
6.613.3 Member Function Documentation . . . . .	2907
6.613.3.1 close . . . . .	2907
6.613.3.2 commit . . . . .	2907
6.613.3.3 createBrowser . . . . .	2908
6.613.3.4 createBrowser . . . . .	2908
6.613.3.5 createBytesMessage . . . . .	2909
6.613.3.6 createBytesMessage . . . . .	2909
6.613.3.7 createCachedConsumer . . . . .	2909
6.613.3.8 createCachedProducer . . . . .	2910
6.613.3.9 createConsumer . . . . .	2910
6.613.3.10createConsumer . . . . .	2911
6.613.3.11createConsumer . . . . .	2911
6.613.3.12createDurableConsumer . . . . .	2912
6.613.3.13createMapMessage . . . . .	2913
6.613.3.14createMessage . . . . .	2913
6.613.3.15createProducer . . . . .	2913
6.613.3.16createQueue . . . . .	2914
6.613.3.17createStreamMessage . . . . .	2914
6.613.3.18createTemporaryQueue . . . . .	2914
6.613.3.19createTemporaryTopic . . . . .	2915
6.613.3.20createTextMessage . . . . .	2915
6.613.3.21createTextMessage . . . . .	2915
6.613.3.22createTopic . . . . .	2915
6.613.3.23getAcknowledgeMode . . . . .	2916
6.613.3.24getSession . . . . .	2916
6.613.3.25getSession . . . . .	2916
6.613.3.26sTransacted . . . . .	2916
6.613.3.27operator= . . . . .	2917
6.613.3.28recover . . . . .	2917
6.613.3.29rollback . . . . .	2917

6.613.3.30unsubscribe . . . . .	2918
6.614decaf::util::concurrent::PooledThread Class Reference . . . . .	2918
6.614.1 Constructor & Destructor Documentation . . . . .	2919
6.614.1.1 PooledThread . . . . .	2919
6.614.1.2 ~PooledThread . . . . .	2919
6.614.2 Member Function Documentation . . . . .	2919
6.614.2.1 getPooledThreadListener . . . . .	2919
6.614.2.2 isBusy . . . . .	2919
6.614.2.3 run . . . . .	2919
6.614.2.4 setPooledThreadListener . . . . .	2920
6.614.2.5 stop . . . . .	2920
6.615decaf::util::concurrent::PooledThreadListener Class Reference . . . . .	2920
6.615.1 Detailed Description . . . . .	2921
6.615.2 Constructor & Destructor Documentation . . . . .	2921
6.615.2.1 ~PooledThreadListener . . . . .	2921
6.615.3 Member Function Documentation . . . . .	2921
6.615.3.1 onTaskCompleted . . . . .	2921
6.615.3.2 onTaskException . . . . .	2921
6.615.3.3 onTaskStarted . . . . .	2922
6.616decaf::net::PortUnreachableException Class Reference . . . . .	2922
6.616.1 Constructor & Destructor Documentation . . . . .	2923
6.616.1.1 PortUnreachableException . . . . .	2923
6.616.1.2 PortUnreachableException . . . . .	2923
6.616.1.3 PortUnreachableException . . . . .	2923
6.616.1.4 PortUnreachableException . . . . .	2923
6.616.1.5 PortUnreachableException . . . . .	2923
6.616.1.6 PortUnreachableException . . . . .	2924
6.616.1.7 ~PortUnreachableException . . . . .	2924
6.616.2 Member Function Documentation . . . . .	2924
6.616.2.1 clone . . . . .	2924
6.617activemq::core::PrefetchPolicy Class Reference . . . . .	2924
6.617.1 Detailed Description . . . . .	2925
6.617.2 Constructor & Destructor Documentation . . . . .	2926
6.617.2.1 PrefetchPolicy . . . . .	2926

6.617.2.2 ~PrefetchPolicy . . . . .	2926
6.617.3 Member Function Documentation . . . . .	2926
6.617.3.1 clone . . . . .	2926
6.617.3.2 configure . . . . .	2926
6.617.3.3 getDurableTopicPrefetch . . . . .	2926
6.617.3.4 getMaxPrefetchLimit . . . . .	2927
6.617.3.5 getQueueBrowserPrefetch . . . . .	2927
6.617.3.6 getQueuePrefetch . . . . .	2927
6.617.3.7 getTopicPrefetch . . . . .	2927
6.617.3.8 setDurableTopicPrefetch . . . . .	2928
6.617.3.9 setQueueBrowserPrefetch . . . . .	2928
6.617.3.10setQueuePrefetch . . . . .	2928
6.617.3.11setTopicPrefetch . . . . .	2928
6.618activemq::util::PrimitiveList Class Reference . . . . .	2929
6.618.1 Detailed Description . . . . .	2931
6.618.2 Constructor & Destructor Documentation . . . . .	2931
6.618.2.1 PrimitiveList . . . . .	2931
6.618.2.2 ~PrimitiveList . . . . .	2931
6.618.2.3 PrimitiveList . . . . .	2931
6.618.2.4 PrimitiveList . . . . .	2931
6.618.3 Member Function Documentation . . . . .	2931
6.618.3.1 getBool . . . . .	2932
6.618.3.2 getByte . . . . .	2932
6.618.3.3 getByteArray . . . . .	2933
6.618.3.4 getChar . . . . .	2933
6.618.3.5 getDouble . . . . .	2934
6.618.3.6 getFloat . . . . .	2934
6.618.3.7 getInt . . . . .	2935
6.618.3.8 getLong . . . . .	2935
6.618.3.9 getShort . . . . .	2936
6.618.3.10getString . . . . .	2936
6.618.3.11setBool . . . . .	2937
6.618.3.12setByte . . . . .	2937
6.618.3.13setByteArray . . . . .	2937

6.618.3.14	setChar . . . . .	2938
6.618.3.15	setDouble . . . . .	2938
6.618.3.16	setFloat . . . . .	2938
6.618.3.17	setInt . . . . .	2939
6.618.3.18	setLong . . . . .	2939
6.618.3.19	setShort . . . . .	2940
6.618.3.20	setString . . . . .	2940
6.618.3.21	toString . . . . .	2940
6.619	activemq::util::PrimitiveMap Class Reference . . . . .	2941
6.619.1	Detailed Description . . . . .	2942
6.619.2	Constructor & Destructor Documentation . . . . .	2943
6.619.2.1	PrimitiveMap . . . . .	2943
6.619.2.2	~PrimitiveMap . . . . .	2943
6.619.2.3	PrimitiveMap . . . . .	2943
6.619.2.4	PrimitiveMap . . . . .	2943
6.619.3	Member Function Documentation . . . . .	2943
6.619.3.1	getBool . . . . .	2943
6.619.3.2	getByte . . . . .	2944
6.619.3.3	getByteArray . . . . .	2944
6.619.3.4	getChar . . . . .	2945
6.619.3.5	getDouble . . . . .	2945
6.619.3.6	getFloat . . . . .	2946
6.619.3.7	getInt . . . . .	2946
6.619.3.8	getLong . . . . .	2947
6.619.3.9	getShort . . . . .	2947
6.619.3.10	getString . . . . .	2948
6.619.3.11	setBool . . . . .	2948
6.619.3.12	setByte . . . . .	2948
6.619.3.13	setByteArray . . . . .	2949
6.619.3.14	setChar . . . . .	2949
6.619.3.15	setDouble . . . . .	2949
6.619.3.16	setFloat . . . . .	2949
6.619.3.17	setInt . . . . .	2950
6.619.3.18	setLong . . . . .	2950

6.619.3.19	setShort . . . . .	2950
6.619.3.20	setString . . . . .	2950
6.619.3.21	toString . . . . .	2951
6.620	activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference . . . . .	2951
6.620.1	Detailed Description . . . . .	2952
6.620.2	Constructor & Destructor Documentation . . . . .	2952
6.620.2.1	PrimitiveTypesMarshaller . . . . .	2952
6.620.2.2	~PrimitiveTypesMarshaller . . . . .	2953
6.620.3	Member Function Documentation . . . . .	2953
6.620.3.1	marshal . . . . .	2953
6.620.3.2	marshal . . . . .	2953
6.620.3.3	marshalList . . . . .	2953
6.620.3.4	marshalMap . . . . .	2954
6.620.3.5	marshalPrimitive . . . . .	2954
6.620.3.6	marshalPrimitiveList . . . . .	2954
6.620.3.7	marshalPrimitiveMap . . . . .	2955
6.620.3.8	unmarshal . . . . .	2955
6.620.3.9	unmarshal . . . . .	2955
6.620.3.10	unmarshalList . . . . .	2956
6.620.3.11	unmarshalMap . . . . .	2956
6.620.3.12	unmarshalPrimitive . . . . .	2956
6.620.3.13	unmarshalPrimitiveList . . . . .	2957
6.620.3.14	unmarshalPrimitiveMap . . . . .	2957
6.621	activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference . . . . .	2957
6.621.1	Detailed Description . . . . .	2958
6.621.2	Field Documentation . . . . .	2958
6.621.2.1	boolValue . . . . .	2958
6.621.2.2	byteArrayValue . . . . .	2958
6.621.2.3	byteValue . . . . .	2958
6.621.2.4	charValue . . . . .	2958
6.621.2.5	doubleValue . . . . .	2958
6.621.2.6	floatValue . . . . .	2958
6.621.2.7	intValue . . . . .	2958



6.621.2.8 listValue . . . . .	2958
6.621.2.9 longValue . . . . .	2958
6.621.2.10mapValue . . . . .	2958
6.621.2.11shortValue . . . . .	2958
6.621.2.12stringValue . . . . .	2959
6.622activemq::util::PrimitiveValueConverter Class Reference . . . . .	2959
6.622.1 Detailed Description . . . . .	2959
6.622.2 Constructor & Destructor Documentation . . . . .	2959
6.622.2.1 PrimitiveValueConverter . . . . .	2959
6.622.2.2 ~PrimitiveValueConverter . . . . .	2959
6.622.3 Member Function Documentation . . . . .	2960
6.622.3.1 convert . . . . .	2960
6.623activemq::util::PrimitiveValueNode Class Reference . . . . .	2960
6.623.1 Detailed Description . . . . .	2963
6.623.2 Member Enumeration Documentation . . . . .	2963
6.623.2.1 PrimitiveType . . . . .	2963
6.623.3 Constructor & Destructor Documentation . . . . .	2964
6.623.3.1 PrimitiveValueNode . . . . .	2964
6.623.3.2 PrimitiveValueNode . . . . .	2964
6.623.3.3 PrimitiveValueNode . . . . .	2964
6.623.3.4 PrimitiveValueNode . . . . .	2964
6.623.3.5 PrimitiveValueNode . . . . .	2965
6.623.3.6 PrimitiveValueNode . . . . .	2965
6.623.3.7 PrimitiveValueNode . . . . .	2965
6.623.3.8 PrimitiveValueNode . . . . .	2965
6.623.3.9 PrimitiveValueNode . . . . .	2965
6.623.3.10PrimitiveValueNode . . . . .	2965
6.623.3.11PrimitiveValueNode . . . . .	2966
6.623.3.12PrimitiveValueNode . . . . .	2966
6.623.3.13PrimitiveValueNode . . . . .	2966
6.623.3.14PrimitiveValueNode . . . . .	2966
6.623.3.15PrimitiveValueNode . . . . .	2966
6.623.3.16~PrimitiveValueNode . . . . .	2967
6.623.4 Member Function Documentation . . . . .	2967

6.623.4.1 clear . . . . .	2967
6.623.4.2 getBool . . . . .	2967
6.623.4.3 getByte . . . . .	2967
6.623.4.4 getByteArray . . . . .	2967
6.623.4.5 getChar . . . . .	2968
6.623.4.6 getDouble . . . . .	2968
6.623.4.7 getFloat . . . . .	2968
6.623.4.8 getInt . . . . .	2969
6.623.4.9 getList . . . . .	2969
6.623.4.10getLong . . . . .	2969
6.623.4.11getMap . . . . .	2970
6.623.4.12getShort . . . . .	2970
6.623.4.13getString . . . . .	2970
6.623.4.14getType . . . . .	2970
6.623.4.15getValue . . . . .	2971
6.623.4.16operator= . . . . .	2971
6.623.4.17operator== . . . . .	2971
6.623.4.18setBool . . . . .	2971
6.623.4.19setByte . . . . .	2971
6.623.4.20setByteArray . . . . .	2972
6.623.4.21setChar . . . . .	2972
6.623.4.22setDouble . . . . .	2972
6.623.4.23setFloat . . . . .	2972
6.623.4.24setInt . . . . .	2973
6.623.4.25setList . . . . .	2973
6.623.4.26setLong . . . . .	2973
6.623.4.27setMap . . . . .	2973
6.623.4.28setShort . . . . .	2973
6.623.4.29setString . . . . .	2974
6.623.4.30setValue . . . . .	2974
6.623.4.31toString . . . . .	2974
6.624decaf::security::Principal Class Reference . . . . .	2974
6.624.1 Detailed Description . . . . .	2975
6.624.2 Constructor & Destructor Documentation . . . . .	2975

6.624.2.1 ~Principal . . . . .	2975
6.624.3 Member Function Documentation . . . . .	2975
6.624.3.1 equals . . . . .	2975
6.624.3.2 getName . . . . .	2975
6.625decaf::util::PriorityQueue< E > Class Template Reference . . . . .	2975
6.625.1 Detailed Description . . . . .	2977
6.625.2 Constructor & Destructor Documentation . . . . .	2978
6.625.2.1 PriorityQueue . . . . .	2978
6.625.2.2 PriorityQueue . . . . .	2978
6.625.2.3 PriorityQueue . . . . .	2978
6.625.2.4 PriorityQueue . . . . .	2979
6.625.2.5 PriorityQueue . . . . .	2979
6.625.2.6 ~PriorityQueue . . . . .	2979
6.625.3 Member Function Documentation . . . . .	2979
6.625.3.1 add . . . . .	2979
6.625.3.2 clear . . . . .	2980
6.625.3.3 comparator . . . . .	2980
6.625.3.4 iterator . . . . .	2980
6.625.3.5 iterator . . . . .	2980
6.625.3.6 offer . . . . .	2981
6.625.3.7 operator= . . . . .	2981
6.625.3.8 operator= . . . . .	2981
6.625.3.9 peek . . . . .	2982
6.625.3.10poll . . . . .	2982
6.625.3.11remove . . . . .	2982
6.625.3.12remove . . . . .	2983
6.625.3.13size . . . . .	2983
6.625.4 Friends And Related Function Documentation . . . . .	2984
6.625.4.1 PriorityQueueIterator . . . . .	2984
6.626activemq::commands::ProducerAck Class Reference . . . . .	2984
6.626.1 Constructor & Destructor Documentation . . . . .	2985
6.626.1.1 ProducerAck . . . . .	2985
6.626.1.2 ~ProducerAck . . . . .	2985
6.626.2 Member Function Documentation . . . . .	2985

6.626.2.1 cloneDataStructure . . . . .	2985
6.626.2.2 copyDataStructure . . . . .	2985
6.626.2.3 equals . . . . .	2986
6.626.2.4 getDataStructureType . . . . .	2986
6.626.2.5 getProducerId . . . . .	2986
6.626.2.6 getProducerId . . . . .	2986
6.626.2.7 getSize . . . . .	2986
6.626.2.8 isProducerAck . . . . .	2986
6.626.2.9 setProducerId . . . . .	2987
6.626.2.10 setSize . . . . .	2987
6.626.2.11 toString . . . . .	2987
6.626.2.12 visit . . . . .	2987
6.626.3 Field Documentation . . . . .	2987
6.626.3.1 ID_PRODUCERACK . . . . .	2987
6.626.3.2 producerId . . . . .	2987
6.626.3.3 size . . . . .	2987
6.627activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class	
Reference . . . . .	2988
6.627.1 Detailed Description . . . . .	2988
6.627.2 Constructor & Destructor Documentation . . . . .	2989
6.627.2.1 ProducerAckMarshaller . . . . .	2989
6.627.2.2 ~ProducerAckMarshaller . . . . .	2989
6.627.3 Member Function Documentation . . . . .	2989
6.627.3.1 createObject . . . . .	2989
6.627.3.2 getDataStructureType . . . . .	2989
6.627.3.3 looseMarshal . . . . .	2989
6.627.3.4 looseUnmarshal . . . . .	2990
6.627.3.5 tightMarshal1 . . . . .	2990
6.627.3.6 tightMarshal2 . . . . .	2991
6.627.3.7 tightUnmarshal . . . . .	2991
6.628activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class	
Reference . . . . .	2992
6.628.1 Detailed Description . . . . .	2992
6.628.2 Constructor & Destructor Documentation . . . . .	2993

6.628.2.1	ProducerAckMarshaller . . . . .	2993
6.628.2.2	~ProducerAckMarshaller . . . . .	2993
6.628.3	Member Function Documentation . . . . .	2993
6.628.3.1	createObject . . . . .	2993
6.628.3.2	getDataStructureType . . . . .	2993
6.628.3.3	looseMarshal . . . . .	2993
6.628.3.4	looseUnmarshal . . . . .	2994
6.628.3.5	tightMarshal1 . . . . .	2994
6.628.3.6	tightMarshal2 . . . . .	2995
6.628.3.7	tightUnmarshal . . . . .	2995
6.629	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class	
	Reference . . . . .	2996
6.629.1	Detailed Description . . . . .	2996
6.629.2	Constructor & Destructor Documentation . . . . .	2997
6.629.2.1	ProducerAckMarshaller . . . . .	2997
6.629.2.2	~ProducerAckMarshaller . . . . .	2997
6.629.3	Member Function Documentation . . . . .	2997
6.629.3.1	createObject . . . . .	2997
6.629.3.2	getDataStructureType . . . . .	2997
6.629.3.3	looseMarshal . . . . .	2997
6.629.3.4	looseUnmarshal . . . . .	2998
6.629.3.5	tightMarshal1 . . . . .	2998
6.629.3.6	tightMarshal2 . . . . .	2999
6.629.3.7	tightUnmarshal . . . . .	2999
6.630	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class	
	Reference . . . . .	3000
6.630.1	Detailed Description . . . . .	3000
6.630.2	Constructor & Destructor Documentation . . . . .	3001
6.630.2.1	ProducerAckMarshaller . . . . .	3001
6.630.2.2	~ProducerAckMarshaller . . . . .	3001
6.630.3	Member Function Documentation . . . . .	3001
6.630.3.1	createObject . . . . .	3001
6.630.3.2	getDataStructureType . . . . .	3001
6.630.3.3	looseMarshal . . . . .	3001

6.630.3.4 looseUnmarshal . . . . .	3002
6.630.3.5 tightMarshal1 . . . . .	3002
6.630.3.6 tightMarshal2 . . . . .	3003
6.630.3.7 tightUnmarshal . . . . .	3003
6.631activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller Class	
Reference . . . . .	3004
6.631.1 Detailed Description . . . . .	3004
6.631.2 Constructor & Destructor Documentation . . . . .	3005
6.631.2.1 ProducerAckMarshaller . . . . .	3005
6.631.2.2 ~ProducerAckMarshaller . . . . .	3005
6.631.3 Member Function Documentation . . . . .	3005
6.631.3.1 createObject . . . . .	3005
6.631.3.2 getDataStructureType . . . . .	3005
6.631.3.3 looseMarshal . . . . .	3005
6.631.3.4 looseUnmarshal . . . . .	3006
6.631.3.5 tightMarshal1 . . . . .	3006
6.631.3.6 tightMarshal2 . . . . .	3007
6.631.3.7 tightUnmarshal . . . . .	3007
6.632activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class	
Reference . . . . .	3008
6.632.1 Detailed Description . . . . .	3008
6.632.2 Constructor & Destructor Documentation . . . . .	3009
6.632.2.1 ProducerAckMarshaller . . . . .	3009
6.632.2.2 ~ProducerAckMarshaller . . . . .	3009
6.632.3 Member Function Documentation . . . . .	3009
6.632.3.1 createObject . . . . .	3009
6.632.3.2 getDataStructureType . . . . .	3009
6.632.3.3 looseMarshal . . . . .	3009
6.632.3.4 looseUnmarshal . . . . .	3010
6.632.3.5 tightMarshal1 . . . . .	3010
6.632.3.6 tightMarshal2 . . . . .	3011
6.632.3.7 tightUnmarshal . . . . .	3011
6.633activemq::cmsutil::ProducerCallback Class Reference . . . . .	3012
6.633.1 Detailed Description . . . . .	3012

6.633.2 Constructor & Destructor Documentation . . . . .	3012
6.633.2.1 ~ProducerCallback . . . . .	3012
6.633.3 Member Function Documentation . . . . .	3012
6.633.3.1 doInCms . . . . .	3012
6.634activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference . .	3013
6.634.1 Constructor & Destructor Documentation . . . . .	3013
6.634.1.1 ProducerExecutor . . . . .	3013
6.634.1.2 ProducerExecutor . . . . .	3013
6.634.1.3 ~ProducerExecutor . . . . .	3013
6.634.2 Member Function Documentation . . . . .	3014
6.634.2.1 doInCms . . . . .	3014
6.634.2.2 getDestination . . . . .	3014
6.634.2.3 operator= . . . . .	3014
6.634.3 Field Documentation . . . . .	3014
6.634.3.1 action . . . . .	3014
6.634.3.2 destination . . . . .	3014
6.634.3.3 parent . . . . .	3014
6.635activemq::commands::ProducerId Class Reference . . . . .	3014
6.635.1 Member Typedef Documentation . . . . .	3016
6.635.1.1 COMPARATOR . . . . .	3016
6.635.2 Constructor & Destructor Documentation . . . . .	3016
6.635.2.1 ProducerId . . . . .	3016
6.635.2.2 ProducerId . . . . .	3016
6.635.2.3 ProducerId . . . . .	3016
6.635.2.4 ProducerId . . . . .	3016
6.635.2.5 ~ProducerId . . . . .	3016
6.635.3 Member Function Documentation . . . . .	3016
6.635.3.1 cloneDataStructure . . . . .	3016
6.635.3.2 compareTo . . . . .	3016
6.635.3.3 copyDataStructure . . . . .	3016
6.635.3.4 equals . . . . .	3017
6.635.3.5 equals . . . . .	3017
6.635.3.6 getConnectionId . . . . .	3017
6.635.3.7 getConnectionId . . . . .	3017

6.635.3.8	getDataStructureType	3017
6.635.3.9	getParentId	3017
6.635.3.10	getSessionId	3017
6.635.3.11	getValue	3017
6.635.3.12	operator<	3018
6.635.3.13	operator=	3018
6.635.3.14	operator==	3018
6.635.3.15	setConnectionId	3018
6.635.3.16	setProducerSessionKey	3018
6.635.3.17	setSessionId	3018
6.635.3.18	setValue	3018
6.635.3.19	toString	3018
6.635.4	Field Documentation	3018
6.635.4.1	connectionId	3018
6.635.4.2	ID_PRODUCERID	3018
6.635.4.3	sessionId	3018
6.635.4.4	value	3018
6.636	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class	
	Reference	3019
6.636.1	Detailed Description	3020
6.636.2	Constructor & Destructor Documentation	3020
6.636.2.1	ProducerIdMarshaller	3020
6.636.2.2	~ProducerIdMarshaller	3020
6.636.3	Member Function Documentation	3020
6.636.3.1	createObject	3020
6.636.3.2	getDataStructureType	3020
6.636.3.3	looseMarshal	3020
6.636.3.4	looseUnmarshal	3021
6.636.3.5	tightMarshal1	3021
6.636.3.6	tightMarshal2	3022
6.636.3.7	tightUnmarshal	3022
6.637	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class	
	Reference	3023
6.637.1	Detailed Description	3024



6.637.2 Constructor & Destructor Documentation . . . . .	3024
6.637.2.1 ProducerIdMarshaller . . . . .	3024
6.637.2.2 ~ProducerIdMarshaller . . . . .	3024
6.637.3 Member Function Documentation . . . . .	3024
6.637.3.1 createObject . . . . .	3024
6.637.3.2 getDataStructureType . . . . .	3024
6.637.3.3 looseMarshal . . . . .	3024
6.637.3.4 looseUnmarshal . . . . .	3025
6.637.3.5 tightMarshal1 . . . . .	3025
6.637.3.6 tightMarshal2 . . . . .	3026
6.637.3.7 tightUnmarshal . . . . .	3026
6.638activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class	
Reference . . . . .	3027
6.638.1 Detailed Description . . . . .	3028
6.638.2 Constructor & Destructor Documentation . . . . .	3028
6.638.2.1 ProducerIdMarshaller . . . . .	3028
6.638.2.2 ~ProducerIdMarshaller . . . . .	3028
6.638.3 Member Function Documentation . . . . .	3028
6.638.3.1 createObject . . . . .	3028
6.638.3.2 getDataStructureType . . . . .	3028
6.638.3.3 looseMarshal . . . . .	3028
6.638.3.4 looseUnmarshal . . . . .	3029
6.638.3.5 tightMarshal1 . . . . .	3029
6.638.3.6 tightMarshal2 . . . . .	3030
6.638.3.7 tightUnmarshal . . . . .	3030
6.639activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class	
Reference . . . . .	3031
6.639.1 Detailed Description . . . . .	3032
6.639.2 Constructor & Destructor Documentation . . . . .	3032
6.639.2.1 ProducerIdMarshaller . . . . .	3032
6.639.2.2 ~ProducerIdMarshaller . . . . .	3032
6.639.3 Member Function Documentation . . . . .	3032
6.639.3.1 createObject . . . . .	3032
6.639.3.2 getDataStructureType . . . . .	3032

6.639.3.3 looseMarshal . . . . .	3032
6.639.3.4 looseUnmarshal . . . . .	3033
6.639.3.5 tightMarshal1 . . . . .	3033
6.639.3.6 tightMarshal2 . . . . .	3034
6.639.3.7 tightUnmarshal . . . . .	3034
6.640activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller Class	
Reference . . . . .	3035
6.640.1 Detailed Description . . . . .	3036
6.640.2 Constructor & Destructor Documentation . . . . .	3036
6.640.2.1 ProducerIdMarshaller . . . . .	3036
6.640.2.2 ~ProducerIdMarshaller . . . . .	3036
6.640.3 Member Function Documentation . . . . .	3036
6.640.3.1 createObject . . . . .	3036
6.640.3.2 getDataStructureType . . . . .	3036
6.640.3.3 looseMarshal . . . . .	3036
6.640.3.4 looseUnmarshal . . . . .	3037
6.640.3.5 tightMarshal1 . . . . .	3037
6.640.3.6 tightMarshal2 . . . . .	3038
6.640.3.7 tightUnmarshal . . . . .	3038
6.641activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class	
Reference . . . . .	3039
6.641.1 Detailed Description . . . . .	3040
6.641.2 Constructor & Destructor Documentation . . . . .	3040
6.641.2.1 ProducerIdMarshaller . . . . .	3040
6.641.2.2 ~ProducerIdMarshaller . . . . .	3040
6.641.3 Member Function Documentation . . . . .	3040
6.641.3.1 createObject . . . . .	3040
6.641.3.2 getDataStructureType . . . . .	3040
6.641.3.3 looseMarshal . . . . .	3040
6.641.3.4 looseUnmarshal . . . . .	3041
6.641.3.5 tightMarshal1 . . . . .	3041
6.641.3.6 tightMarshal2 . . . . .	3042
6.641.3.7 tightUnmarshal . . . . .	3042
6.642activemq::commands::ProducerInfo Class Reference . . . . .	3043

6.642.1 Constructor & Destructor Documentation . . . . .	3044
6.642.1.1 ProducerInfo . . . . .	3044
6.642.1.2 ~ProducerInfo . . . . .	3044
6.642.2 Member Function Documentation . . . . .	3044
6.642.2.1 cloneDataStructure . . . . .	3044
6.642.2.2 copyDataStructure . . . . .	3044
6.642.2.3 createRemoveCommand . . . . .	3045
6.642.2.4 equals . . . . .	3045
6.642.2.5 getBrokerPath . . . . .	3045
6.642.2.6 getBrokerPath . . . . .	3045
6.642.2.7 getDataStructureType . . . . .	3045
6.642.2.8 getDestination . . . . .	3045
6.642.2.9 getDestination . . . . .	3045
6.642.2.10getProducerId . . . . .	3045
6.642.2.11getProducerId . . . . .	3046
6.642.2.12getWindowSize . . . . .	3046
6.642.2.13sDispatchAsync . . . . .	3046
6.642.2.14sProducerInfo . . . . .	3046
6.642.2.15setBrokerPath . . . . .	3046
6.642.2.16setDestination . . . . .	3046
6.642.2.17setDispatchAsync . . . . .	3046
6.642.2.18setProducerId . . . . .	3046
6.642.2.19setWindowSize . . . . .	3046
6.642.2.20toString . . . . .	3046
6.642.2.21visit . . . . .	3047
6.642.3 Field Documentation . . . . .	3047
6.642.3.1 brokerPath . . . . .	3047
6.642.3.2 destination . . . . .	3047
6.642.3.3 dispatchAsync . . . . .	3047
6.642.3.4 ID_PRODUCERINFO . . . . .	3047
6.642.3.5 producerId . . . . .	3047
6.642.3.6 windowSize . . . . .	3047
6.643activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference . . . . .	3047

6.643.1 Detailed Description . . . . .	3048
6.643.2 Constructor & Destructor Documentation . . . . .	3048
6.643.2.1 ProducerInfoMarshaller . . . . .	3048
6.643.2.2 ~ProducerInfoMarshaller . . . . .	3049
6.643.3 Member Function Documentation . . . . .	3049
6.643.3.1 createObject . . . . .	3049
6.643.3.2 getDataStructureType . . . . .	3049
6.643.3.3 looseMarshal . . . . .	3049
6.643.3.4 looseUnmarshal . . . . .	3050
6.643.3.5 tightMarshal1 . . . . .	3050
6.643.3.6 tightMarshal2 . . . . .	3051
6.643.3.7 tightUnmarshal . . . . .	3051
6.644activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class	
Reference . . . . .	3052
6.644.1 Detailed Description . . . . .	3052
6.644.2 Constructor & Destructor Documentation . . . . .	3053
6.644.2.1 ProducerInfoMarshaller . . . . .	3053
6.644.2.2 ~ProducerInfoMarshaller . . . . .	3053
6.644.3 Member Function Documentation . . . . .	3053
6.644.3.1 createObject . . . . .	3053
6.644.3.2 getDataStructureType . . . . .	3053
6.644.3.3 looseMarshal . . . . .	3053
6.644.3.4 looseUnmarshal . . . . .	3054
6.644.3.5 tightMarshal1 . . . . .	3054
6.644.3.6 tightMarshal2 . . . . .	3055
6.644.3.7 tightUnmarshal . . . . .	3055
6.645activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class	
Reference . . . . .	3056
6.645.1 Detailed Description . . . . .	3056
6.645.2 Constructor & Destructor Documentation . . . . .	3057
6.645.2.1 ProducerInfoMarshaller . . . . .	3057
6.645.2.2 ~ProducerInfoMarshaller . . . . .	3057
6.645.3 Member Function Documentation . . . . .	3057
6.645.3.1 createObject . . . . .	3057

6.645.3.2	getDataStructureType . . . . .	3057
6.645.3.3	looseMarshal . . . . .	3057
6.645.3.4	looseUnmarshal . . . . .	3058
6.645.3.5	tightMarshal1 . . . . .	3058
6.645.3.6	tightMarshal2 . . . . .	3059
6.645.3.7	tightUnmarshal . . . . .	3059
6.646	activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class	
	Reference . . . . .	3060
6.646.1	Detailed Description . . . . .	3060
6.646.2	Constructor & Destructor Documentation . . . . .	3061
	6.646.2.1 ProducerInfoMarshaller . . . . .	3061
	6.646.2.2 ~ProducerInfoMarshaller . . . . .	3061
6.646.3	Member Function Documentation . . . . .	3061
	6.646.3.1 createObject . . . . .	3061
	6.646.3.2 getDataStructureType . . . . .	3061
	6.646.3.3 looseMarshal . . . . .	3061
	6.646.3.4 looseUnmarshal . . . . .	3062
	6.646.3.5 tightMarshal1 . . . . .	3062
	6.646.3.6 tightMarshal2 . . . . .	3063
	6.646.3.7 tightUnmarshal . . . . .	3063
6.647	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class	
	Reference . . . . .	3064
6.647.1	Detailed Description . . . . .	3064
6.647.2	Constructor & Destructor Documentation . . . . .	3065
	6.647.2.1 ProducerInfoMarshaller . . . . .	3065
	6.647.2.2 ~ProducerInfoMarshaller . . . . .	3065
6.647.3	Member Function Documentation . . . . .	3065
	6.647.3.1 createObject . . . . .	3065
	6.647.3.2 getDataStructureType . . . . .	3065
	6.647.3.3 looseMarshal . . . . .	3065
	6.647.3.4 looseUnmarshal . . . . .	3066
	6.647.3.5 tightMarshal1 . . . . .	3066
	6.647.3.6 tightMarshal2 . . . . .	3067
	6.647.3.7 tightUnmarshal . . . . .	3067

6.648activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller Class Reference . . . . .	3068
6.648.1 Detailed Description . . . . .	3068
6.648.2 Constructor & Destructor Documentation . . . . .	3069
6.648.2.1 ProducerInfoMarshaller . . . . .	3069
6.648.2.2 ~ProducerInfoMarshaller . . . . .	3069
6.648.3 Member Function Documentation . . . . .	3069
6.648.3.1 createObject . . . . .	3069
6.648.3.2 getDataStructureType . . . . .	3069
6.648.3.3 looseMarshal . . . . .	3069
6.648.3.4 looseUnmarshal . . . . .	3070
6.648.3.5 tightMarshal1 . . . . .	3070
6.648.3.6 tightMarshal2 . . . . .	3071
6.648.3.7 tightUnmarshal . . . . .	3071
6.649activemq::state::ProducerState Class Reference . . . . .	3072
6.649.1 Constructor & Destructor Documentation . . . . .	3072
6.649.1.1 ProducerState . . . . .	3072
6.649.1.2 ~ProducerState . . . . .	3072
6.649.2 Member Function Documentation . . . . .	3072
6.649.2.1 getInfo . . . . .	3072
6.649.2.2 getTransactionState . . . . .	3072
6.649.2.3 setTransactionState . . . . .	3072
6.649.2.4 toString . . . . .	3072
6.650decaf::util::Properties Class Reference . . . . .	3072
6.650.1 Detailed Description . . . . .	3074
6.650.2 Constructor & Destructor Documentation . . . . .	3074
6.650.2.1 Properties . . . . .	3074
6.650.2.2 Properties . . . . .	3074
6.650.2.3 ~Properties . . . . .	3074
6.650.3 Member Function Documentation . . . . .	3074
6.650.3.1 clear . . . . .	3074
6.650.3.2 clone . . . . .	3075
6.650.3.3 copy . . . . .	3075
6.650.3.4 equals . . . . .	3075

6.650.3.5	getProperty	3075
6.650.3.6	getProperty	3076
6.650.3.7	hasProperty	3076
6.650.3.8	isEmpty	3076
6.650.3.9	load	3076
6.650.3.10	load	3077
6.650.3.11	operator=	3079
6.650.3.12	propertyNames	3079
6.650.3.13	remove	3079
6.650.3.14	setProperty	3079
6.650.3.15	size	3080
6.650.3.16	store	3080
6.650.3.17	store	3081
6.650.3.18	toArray	3081
6.650.3.19	toString	3082
6.650.4	Field Documentation	3082
6.650.4.1	defaults	3082
6.651	decaf::util::logging::PropertiesChangeListener Class Reference	3082
6.651.1	Detailed Description	3082
6.651.2	Constructor & Destructor Documentation	3083
6.651.2.1	~PropertiesChangeListener	3083
6.651.3	Member Function Documentation	3083
6.651.3.1	onPropertiesReset	3083
6.651.3.2	onPropertyChanged	3083
6.652	decaf::net::ProtocolException Class Reference	3083
6.652.1	Constructor & Destructor Documentation	3084
6.652.1.1	ProtocolException	3084
6.652.1.2	ProtocolException	3084
6.652.1.3	ProtocolException	3084
6.652.1.4	ProtocolException	3084
6.652.1.5	ProtocolException	3085
6.652.1.6	ProtocolException	3085
6.652.1.7	~ProtocolException	3085
6.652.2	Member Function Documentation	3085

6.652.2.1 clone . . . . .	3085
6.653decaf::security::PublicKey Class Reference . . . . .	3086
6.653.1 Detailed Description . . . . .	3086
6.653.2 Constructor & Destructor Documentation . . . . .	3086
6.653.2.1 ~PublicKey . . . . .	3086
6.654decaf::io::PushbackInputStream Class Reference . . . . .	3086
6.654.1 Detailed Description . . . . .	3088
6.654.2 Constructor & Destructor Documentation . . . . .	3088
6.654.2.1 PushbackInputStream . . . . .	3088
6.654.2.2 PushbackInputStream . . . . .	3089
6.654.2.3 ~PushbackInputStream . . . . .	3089
6.654.3 Member Function Documentation . . . . .	3089
6.654.3.1 available . . . . .	3089
6.654.3.2 doReadArrayBounded . . . . .	3090
6.654.3.3 doReadByte . . . . .	3090
6.654.3.4 mark . . . . .	3090
6.654.3.5 markSupported . . . . .	3090
6.654.3.6 reset . . . . .	3091
6.654.3.7 skip . . . . .	3091
6.654.3.8 unread . . . . .	3092
6.654.3.9 unread . . . . .	3092
6.654.3.10unread . . . . .	3093
6.655cms::Queue Class Reference . . . . .	3093
6.655.1 Detailed Description . . . . .	3094
6.655.2 Constructor & Destructor Documentation . . . . .	3094
6.655.2.1 ~Queue . . . . .	3094
6.655.3 Member Function Documentation . . . . .	3094
6.655.3.1 getQueueName . . . . .	3094
6.656decaf::util::Queue< E > Class Template Reference . . . . .	3094
6.656.1 Detailed Description . . . . .	3095
6.656.2 Constructor & Destructor Documentation . . . . .	3095
6.656.2.1 ~Queue . . . . .	3096
6.656.3 Member Function Documentation . . . . .	3096
6.656.3.1 element . . . . .	3096



6.656.3.2 offer . . . . .	3096
6.656.3.3 peek . . . . .	3097
6.656.3.4 poll . . . . .	3097
6.656.3.5 remove . . . . .	3098
6.657cms::QueueBrowser Class Reference . . . . .	3098
6.657.1 Detailed Description . . . . .	3098
6.657.2 Constructor & Destructor Documentation . . . . .	3099
6.657.2.1 ~QueueBrowser . . . . .	3099
6.657.3 Member Function Documentation . . . . .	3099
6.657.3.1 getEnumeration . . . . .	3099
6.657.3.2 getMessageSelector . . . . .	3099
6.657.3.3 getQueue . . . . .	3100
6.658decaf::util::Random Class Reference . . . . .	3100
6.658.1 Detailed Description . . . . .	3101
6.658.2 Constructor & Destructor Documentation . . . . .	3102
6.658.2.1 Random . . . . .	3102
6.658.2.2 Random . . . . .	3102
6.658.3 Member Function Documentation . . . . .	3102
6.658.3.1 next . . . . .	3102
6.658.3.2 nextBoolean . . . . .	3103
6.658.3.3 nextBytes . . . . .	3103
6.658.3.4 nextBytes . . . . .	3103
6.658.3.5 nextDouble . . . . .	3104
6.658.3.6 nextFloat . . . . .	3104
6.658.3.7 nextGaussian . . . . .	3104
6.658.3.8 nextInt . . . . .	3104
6.658.3.9 nextInt . . . . .	3105
6.658.3.10nextLong . . . . .	3105
6.658.3.11setSeed . . . . .	3105
6.659decaf::lang::Readable Class Reference . . . . .	3106
6.659.1 Detailed Description . . . . .	3106
6.659.2 Constructor & Destructor Documentation . . . . .	3106
6.659.2.1 ~Readable . . . . .	3106
6.659.3 Member Function Documentation . . . . .	3107

6.659.3.1 read . . . . .	3107
6.660activemq::transport::inactivity::ReadChecker Class Reference . . . . .	3107
6.660.1 Detailed Description . . . . .	3108
6.660.2 Constructor & Destructor Documentation . . . . .	3108
6.660.2.1 ReadChecker . . . . .	3108
6.660.2.2 ~ReadChecker . . . . .	3108
6.660.3 Member Function Documentation . . . . .	3108
6.660.3.1 run . . . . .	3108
6.661decaf::io::Reader Class Reference . . . . .	3108
6.661.1 Constructor & Destructor Documentation . . . . .	3110
6.661.1.1 Reader . . . . .	3110
6.661.1.2 ~Reader . . . . .	3110
6.661.2 Member Function Documentation . . . . .	3110
6.661.2.1 doReadArray . . . . .	3110
6.661.2.2 doReadArrayBounded . . . . .	3110
6.661.2.3 doReadChar . . . . .	3110
6.661.2.4 doReadCharBuffer . . . . .	3110
6.661.2.5 doReadVector . . . . .	3110
6.661.2.6 mark . . . . .	3111
6.661.2.7 markSupported . . . . .	3111
6.661.2.8 read . . . . .	3111
6.661.2.9 read . . . . .	3112
6.661.2.10read . . . . .	3112
6.661.2.11read . . . . .	3113
6.661.2.12read . . . . .	3113
6.661.2.13ready . . . . .	3113
6.661.2.14reset . . . . .	3114
6.661.2.15skip . . . . .	3114
6.662decaf::nio::ReadOnlyBufferException Class Reference . . . . .	3115
6.662.1 Constructor & Destructor Documentation . . . . .	3115
6.662.1.1 ReadOnlyBufferException . . . . .	3115
6.662.1.2 ReadOnlyBufferException . . . . .	3115
6.662.1.3 ReadOnlyBufferException . . . . .	3116
6.662.1.4 ReadOnlyBufferException . . . . .	3116

6.662.1.5	ReadOnlyBufferException . . . . .	3116
6.662.1.6	ReadOnlyBufferException . . . . .	3116
6.662.1.7	~ReadOnlyBufferException . . . . .	3117
6.662.2	Member Function Documentation . . . . .	3117
6.662.2.1	clone . . . . .	3117
6.663	decaf::util::concurrent::locks::ReadWriteLock Class Reference . . . . .	3117
6.663.1	Detailed Description . . . . .	3117
6.663.2	Constructor & Destructor Documentation . . . . .	3118
6.663.2.1	~ReadWriteLock . . . . .	3119
6.663.3	Member Function Documentation . . . . .	3119
6.663.3.1	readLock . . . . .	3119
6.663.3.2	writeLock . . . . .	3119
6.664	activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference . . . . .	3119
6.664.1	Constructor & Destructor Documentation . . . . .	3120
6.664.1.1	ReceiveExecutor . . . . .	3120
6.664.1.2	ReceiveExecutor . . . . .	3120
6.664.1.3	~ReceiveExecutor . . . . .	3120
6.664.2	Member Function Documentation . . . . .	3120
6.664.2.1	doInCms . . . . .	3120
6.664.2.2	getDestination . . . . .	3121
6.664.2.3	getMessage . . . . .	3121
6.664.2.4	operator= . . . . .	3121
6.664.3	Field Documentation . . . . .	3121
6.664.3.1	destination . . . . .	3121
6.664.3.2	message . . . . .	3121
6.664.3.3	noLocal . . . . .	3121
6.664.3.4	parent . . . . .	3121
6.664.3.5	selector . . . . .	3121
6.665	activemq::core::RedeliveryPolicy Class Reference . . . . .	3121
6.665.1	Detailed Description . . . . .	3122
6.665.2	Constructor & Destructor Documentation . . . . .	3123
6.665.2.1	RedeliveryPolicy . . . . .	3123
6.665.2.2	~RedeliveryPolicy . . . . .	3123
6.665.3	Member Function Documentation . . . . .	3123

6.665.3.1 clone . . . . .	3123
6.665.3.2 configure . . . . .	3123
6.665.3.3 getBackOffMultiplier . . . . .	3123
6.665.3.4 getCollisionAvoidancePercent . . . . .	3124
6.665.3.5 getInitialRedeliveryDelay . . . . .	3124
6.665.3.6 getMaximumRedeliveries . . . . .	3124
6.665.3.7 getRedeliveryDelay . . . . .	3124
6.665.3.8 isUseCollisionAvoidance . . . . .	3125
6.665.3.9 isUseExponentialBackOff . . . . .	3125
6.665.3.10setBackOffMultiplier . . . . .	3125
6.665.3.11setCollisionAvoidancePercent . . . . .	3125
6.665.3.12setInitialRedeliveryDelay . . . . .	3125
6.665.3.13setMaximumRedeliveries . . . . .	3126
6.665.3.14setUseCollisionAvoidance . . . . .	3126
6.665.3.15setUseExponentialBackOff . . . . .	3126
6.665.4 Field Documentation . . . . .	3126
6.665.4.1 NO_MAXIMUM_REDELIVERIES . . . . .	3126
6.666decaf::util::concurrent::locks::ReentrantLock Class Reference . . . . .	3126
6.666.1 Detailed Description . . . . .	3127
6.666.2 Constructor & Destructor Documentation . . . . .	3128
6.666.2.1 ReentrantLock . . . . .	3128
6.666.2.2 ~ReentrantLock . . . . .	3128
6.666.3 Member Function Documentation . . . . .	3128
6.666.3.1 getHoldCount . . . . .	3128
6.666.3.2 isFair . . . . .	3129
6.666.3.3 isHeldByCurrentThread . . . . .	3129
6.666.3.4 isLocked . . . . .	3129
6.666.3.5 lock . . . . .	3130
6.666.3.6 lockInterruptibly . . . . .	3130
6.666.3.7 newCondition . . . . .	3131
6.666.3.8 toString . . . . .	3131
6.666.3.9 tryLock . . . . .	3131
6.666.3.10tryLock . . . . .	3133
6.666.3.11unlock . . . . .	3133

6.667decaf::util::concurrent::RejectedExecutionException Class Reference . . .	3134
6.667.1 Constructor & Destructor Documentation . . . . .	3134
6.667.1.1 RejectedExecutionException . . . . .	3134
6.667.1.2 RejectedExecutionException . . . . .	3134
6.667.1.3 RejectedExecutionException . . . . .	3135
6.667.1.4 RejectedExecutionException . . . . .	3135
6.667.1.5 RejectedExecutionException . . . . .	3135
6.667.1.6 RejectedExecutionException . . . . .	3135
6.667.1.7 ~RejectedExecutionException . . . . .	3136
6.667.2 Member Function Documentation . . . . .	3136
6.667.2.1 clone . . . . .	3136
6.668decaf::util::concurrent::RejectedExecutionHandler Class Reference . . .	3136
6.668.1 Detailed Description . . . . .	3137
6.668.2 Constructor & Destructor Documentation . . . . .	3137
6.668.2.1 ~RejectedExecutionHandler . . . . .	3137
6.668.3 Member Function Documentation . . . . .	3137
6.668.3.1 rejectedExecution . . . . .	3137
6.669activemq::commands::RemoveInfo Class Reference . . . . .	3137
6.669.1 Constructor & Destructor Documentation . . . . .	3138
6.669.1.1 RemoveInfo . . . . .	3138
6.669.1.2 ~RemoveInfo . . . . .	3138
6.669.2 Member Function Documentation . . . . .	3139
6.669.2.1 cloneDataStructure . . . . .	3139
6.669.2.2 copyDataStructure . . . . .	3139
6.669.2.3 equals . . . . .	3139
6.669.2.4 getDataStructureType . . . . .	3139
6.669.2.5 getLastDeliveredSequenceId . . . . .	3140
6.669.2.6 getObjectId . . . . .	3140
6.669.2.7 getObjectId . . . . .	3140
6.669.2.8 isRemoveInfo . . . . .	3140
6.669.2.9 setLastDeliveredSequenceId . . . . .	3140
6.669.2.10 setObjectId . . . . .	3140
6.669.2.11 toString . . . . .	3140
6.669.2.12 visit . . . . .	3140

6.669.3 Field Documentation . . . . .	3141
6.669.3.1 ID_REMOVEINFO . . . . .	3141
6.669.3.2 lastDeliveredSequenceId . . . . .	3141
6.669.3.3 objectId . . . . .	3141
6.670activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class	
Reference . . . . .	3141
6.670.1 Detailed Description . . . . .	3142
6.670.2 Constructor & Destructor Documentation . . . . .	3142
6.670.2.1 RemoveInfoMarshaller . . . . .	3142
6.670.2.2 ~RemoveInfoMarshaller . . . . .	3142
6.670.3 Member Function Documentation . . . . .	3142
6.670.3.1 createObject . . . . .	3142
6.670.3.2 getDataStructureType . . . . .	3143
6.670.3.3 looseMarshal . . . . .	3143
6.670.3.4 looseUnmarshal . . . . .	3143
6.670.3.5 tightMarshal1 . . . . .	3144
6.670.3.6 tightMarshal2 . . . . .	3144
6.670.3.7 tightUnmarshal . . . . .	3145
6.671activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller Class	
Reference . . . . .	3145
6.671.1 Detailed Description . . . . .	3146
6.671.2 Constructor & Destructor Documentation . . . . .	3146
6.671.2.1 RemoveInfoMarshaller . . . . .	3146
6.671.2.2 ~RemoveInfoMarshaller . . . . .	3146
6.671.3 Member Function Documentation . . . . .	3146
6.671.3.1 createObject . . . . .	3146
6.671.3.2 getDataStructureType . . . . .	3147
6.671.3.3 looseMarshal . . . . .	3147
6.671.3.4 looseUnmarshal . . . . .	3147
6.671.3.5 tightMarshal1 . . . . .	3148
6.671.3.6 tightMarshal2 . . . . .	3148
6.671.3.7 tightUnmarshal . . . . .	3149
6.672activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class	
Reference . . . . .	3149
6.672.1 Detailed Description . . . . .	3150

6.672.2 Constructor & Destructor Documentation . . . . .	3150
6.672.2.1 RemoveInfoMarshaller . . . . .	3150
6.672.2.2 ~RemoveInfoMarshaller . . . . .	3150
6.672.3 Member Function Documentation . . . . .	3150
6.672.3.1 createObject . . . . .	3150
6.672.3.2 getDataStructureType . . . . .	3151
6.672.3.3 looseMarshal . . . . .	3151
6.672.3.4 looseUnmarshal . . . . .	3151
6.672.3.5 tightMarshal1 . . . . .	3152
6.672.3.6 tightMarshal2 . . . . .	3152
6.672.3.7 tightUnmarshal . . . . .	3153
6.673activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class	
Reference . . . . .	3153
6.673.1 Detailed Description . . . . .	3154
6.673.2 Constructor & Destructor Documentation . . . . .	3154
6.673.2.1 RemoveInfoMarshaller . . . . .	3154
6.673.2.2 ~RemoveInfoMarshaller . . . . .	3154
6.673.3 Member Function Documentation . . . . .	3154
6.673.3.1 createObject . . . . .	3154
6.673.3.2 getDataStructureType . . . . .	3155
6.673.3.3 looseMarshal . . . . .	3155
6.673.3.4 looseUnmarshal . . . . .	3155
6.673.3.5 tightMarshal1 . . . . .	3156
6.673.3.6 tightMarshal2 . . . . .	3156
6.673.3.7 tightUnmarshal . . . . .	3157
6.674activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class	
Reference . . . . .	3157
6.674.1 Detailed Description . . . . .	3158
6.674.2 Constructor & Destructor Documentation . . . . .	3158
6.674.2.1 RemoveInfoMarshaller . . . . .	3158
6.674.2.2 ~RemoveInfoMarshaller . . . . .	3158
6.674.3 Member Function Documentation . . . . .	3158
6.674.3.1 createObject . . . . .	3158
6.674.3.2 getDataStructureType . . . . .	3159

6.674.3.3 looseMarshal . . . . .	3159
6.674.3.4 looseUnmarshal . . . . .	3159
6.674.3.5 tightMarshal1 . . . . .	3160
6.674.3.6 tightMarshal2 . . . . .	3160
6.674.3.7 tightUnmarshal . . . . .	3161
6.675activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference . . . . .	3161
6.675.1 Detailed Description . . . . .	3162
6.675.2 Constructor & Destructor Documentation . . . . .	3162
6.675.2.1 RemoveInfoMarshaller . . . . .	3162
6.675.2.2 ~RemoveInfoMarshaller . . . . .	3162
6.675.3 Member Function Documentation . . . . .	3162
6.675.3.1 createObject . . . . .	3162
6.675.3.2 getDataStructureType . . . . .	3163
6.675.3.3 looseMarshal . . . . .	3163
6.675.3.4 looseUnmarshal . . . . .	3163
6.675.3.5 tightMarshal1 . . . . .	3164
6.675.3.6 tightMarshal2 . . . . .	3164
6.675.3.7 tightUnmarshal . . . . .	3165
6.676activemq::commands::RemoveSubscriptionInfo Class Reference . . . . .	3165
6.676.1 Constructor & Destructor Documentation . . . . .	3166
6.676.1.1 RemoveSubscriptionInfo . . . . .	3166
6.676.1.2 ~RemoveSubscriptionInfo . . . . .	3166
6.676.2 Member Function Documentation . . . . .	3166
6.676.2.1 cloneDataStructure . . . . .	3166
6.676.2.2 copyDataStructure . . . . .	3167
6.676.2.3 equals . . . . .	3167
6.676.2.4 getClientId . . . . .	3167
6.676.2.5 getClientId . . . . .	3167
6.676.2.6 getConnectionId . . . . .	3167
6.676.2.7 getConnectionId . . . . .	3167
6.676.2.8 getDataStructureType . . . . .	3168
6.676.2.9 getSubscriptionName . . . . .	3168
6.676.2.10getSubscriptionName . . . . .	3168



6.676.2.11	isRemoveSubscriptionInfo . . . . .	3168
6.676.2.12	setClientId . . . . .	3168
6.676.2.13	setConnectionId . . . . .	3168
6.676.2.14	setSubscriptionName . . . . .	3168
6.676.2.15	toString . . . . .	3168
6.676.2.16	visit . . . . .	3169
6.676.3	Field Documentation . . . . .	3169
6.676.3.1	clientId . . . . .	3169
6.676.3.2	connectionId . . . . .	3169
6.676.3.3	ID_REMOVESUBSCRIPTIONINFO . . . . .	3169
6.676.3.4	subscriptionName . . . . .	3169
6.677	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	
	Class Reference . . . . .	3169
6.677.1	Detailed Description . . . . .	3170
6.677.2	Constructor & Destructor Documentation . . . . .	3170
6.677.2.1	RemoveSubscriptionInfoMarshaller . . . . .	3170
6.677.2.2	~RemoveSubscriptionInfoMarshaller . . . . .	3170
6.677.3	Member Function Documentation . . . . .	3170
6.677.3.1	createObject . . . . .	3171
6.677.3.2	getDataStructureType . . . . .	3171
6.677.3.3	looseMarshal . . . . .	3171
6.677.3.4	looseUnmarshal . . . . .	3172
6.677.3.5	tightMarshal1 . . . . .	3172
6.677.3.6	tightMarshal2 . . . . .	3173
6.677.3.7	tightUnmarshal . . . . .	3173
6.678	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	
	Class Reference . . . . .	3174
6.678.1	Detailed Description . . . . .	3174
6.678.2	Constructor & Destructor Documentation . . . . .	3175
6.678.2.1	RemoveSubscriptionInfoMarshaller . . . . .	3175
6.678.2.2	~RemoveSubscriptionInfoMarshaller . . . . .	3175
6.678.3	Member Function Documentation . . . . .	3175
6.678.3.1	createObject . . . . .	3175
6.678.3.2	getDataStructureType . . . . .	3175

6.678.3.3 looseMarshal . . . . .	3175
6.678.3.4 looseUnmarshal . . . . .	3176
6.678.3.5 tightMarshal1 . . . . .	3176
6.678.3.6 tightMarshal2 . . . . .	3177
6.678.3.7 tightUnmarshal . . . . .	3177
6.679activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	
Class Reference . . . . .	3178
6.679.1 Detailed Description . . . . .	3178
6.679.2 Constructor & Destructor Documentation . . . . .	3179
6.679.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3179
6.679.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3179
6.679.3 Member Function Documentation . . . . .	3179
6.679.3.1 createObject . . . . .	3179
6.679.3.2 getDataStructureType . . . . .	3179
6.679.3.3 looseMarshal . . . . .	3179
6.679.3.4 looseUnmarshal . . . . .	3180
6.679.3.5 tightMarshal1 . . . . .	3180
6.679.3.6 tightMarshal2 . . . . .	3181
6.679.3.7 tightUnmarshal . . . . .	3181
6.680activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	
Class Reference . . . . .	3182
6.680.1 Detailed Description . . . . .	3182
6.680.2 Constructor & Destructor Documentation . . . . .	3183
6.680.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3183
6.680.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3183
6.680.3 Member Function Documentation . . . . .	3183
6.680.3.1 createObject . . . . .	3183
6.680.3.2 getDataStructureType . . . . .	3183
6.680.3.3 looseMarshal . . . . .	3183
6.680.3.4 looseUnmarshal . . . . .	3184
6.680.3.5 tightMarshal1 . . . . .	3184
6.680.3.6 tightMarshal2 . . . . .	3185
6.680.3.7 tightUnmarshal . . . . .	3185
6.681activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	
Class Reference . . . . .	3186

6.681.1 Detailed Description . . . . .	3186
6.681.2 Constructor & Destructor Documentation . . . . .	3187
6.681.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3187
6.681.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3187
6.681.3 Member Function Documentation . . . . .	3187
6.681.3.1 createObject . . . . .	3187
6.681.3.2 getDataStructureType . . . . .	3187
6.681.3.3 looseMarshal . . . . .	3187
6.681.3.4 looseUnmarshal . . . . .	3188
6.681.3.5 tightMarshal1 . . . . .	3188
6.681.3.6 tightMarshal2 . . . . .	3189
6.681.3.7 tightUnmarshal . . . . .	3189
6.682activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller Class Reference . . . . .	3190
6.682.1 Detailed Description . . . . .	3190
6.682.2 Constructor & Destructor Documentation . . . . .	3191
6.682.2.1 RemoveSubscriptionInfoMarshaller . . . . .	3191
6.682.2.2 ~RemoveSubscriptionInfoMarshaller . . . . .	3191
6.682.3 Member Function Documentation . . . . .	3191
6.682.3.1 createObject . . . . .	3191
6.682.3.2 getDataStructureType . . . . .	3191
6.682.3.3 looseMarshal . . . . .	3191
6.682.3.4 looseUnmarshal . . . . .	3192
6.682.3.5 tightMarshal1 . . . . .	3192
6.682.3.6 tightMarshal2 . . . . .	3193
6.682.3.7 tightUnmarshal . . . . .	3193
6.683activemq::commands::ReplayCommand Class Reference . . . . .	3194
6.683.1 Constructor & Destructor Documentation . . . . .	3195
6.683.1.1 ReplayCommand . . . . .	3195
6.683.1.2 ~ReplayCommand . . . . .	3195
6.683.2 Member Function Documentation . . . . .	3195
6.683.2.1 cloneDataStructure . . . . .	3195
6.683.2.2 copyDataStructure . . . . .	3195
6.683.2.3 equals . . . . .	3195

6.683.2.4	getDataStructureType . . . . .	3196
6.683.2.5	getFirstNakNumber . . . . .	3196
6.683.2.6	getLastNakNumber . . . . .	3196
6.683.2.7	setFirstNakNumber . . . . .	3196
6.683.2.8	setLastNakNumber . . . . .	3196
6.683.2.9	toString . . . . .	3196
6.683.2.10	visit . . . . .	3196
6.683.3	Field Documentation . . . . .	3197
6.683.3.1	firstNakNumber . . . . .	3197
6.683.3.2	ID_REPLAYCOMMAND . . . . .	3197
6.683.3.3	lastNakNumber . . . . .	3197
6.684	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller	
	Class Reference . . . . .	3197
6.684.1	Detailed Description . . . . .	3198
6.684.2	Constructor & Destructor Documentation . . . . .	3198
6.684.2.1	ReplayCommandMarshaller . . . . .	3198
6.684.2.2	~ReplayCommandMarshaller . . . . .	3198
6.684.3	Member Function Documentation . . . . .	3198
6.684.3.1	createObject . . . . .	3198
6.684.3.2	getDataStructureType . . . . .	3198
6.684.3.3	looseMarshal . . . . .	3199
6.684.3.4	looseUnmarshal . . . . .	3199
6.684.3.5	tightMarshal1 . . . . .	3199
6.684.3.6	tightMarshal2 . . . . .	3200
6.684.3.7	tightUnmarshal . . . . .	3200
6.685	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	
	Class Reference . . . . .	3201
6.685.1	Detailed Description . . . . .	3202
6.685.2	Constructor & Destructor Documentation . . . . .	3202
6.685.2.1	ReplayCommandMarshaller . . . . .	3202
6.685.2.2	~ReplayCommandMarshaller . . . . .	3202
6.685.3	Member Function Documentation . . . . .	3202
6.685.3.1	createObject . . . . .	3202
6.685.3.2	getDataStructureType . . . . .	3202

6.685.3.3 looseMarshal . . . . .	3203
6.685.3.4 looseUnmarshal . . . . .	3203
6.685.3.5 tightMarshal1 . . . . .	3203
6.685.3.6 tightMarshal2 . . . . .	3204
6.685.3.7 tightUnmarshal . . . . .	3204
6.686activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	
Class Reference . . . . .	3205
6.686.1 Detailed Description . . . . .	3206
6.686.2 Constructor & Destructor Documentation . . . . .	3206
6.686.2.1 ReplayCommandMarshaller . . . . .	3206
6.686.2.2 ~ReplayCommandMarshaller . . . . .	3206
6.686.3 Member Function Documentation . . . . .	3206
6.686.3.1 createObject . . . . .	3206
6.686.3.2 getDataStructureType . . . . .	3206
6.686.3.3 looseMarshal . . . . .	3207
6.686.3.4 looseUnmarshal . . . . .	3207
6.686.3.5 tightMarshal1 . . . . .	3207
6.686.3.6 tightMarshal2 . . . . .	3208
6.686.3.7 tightUnmarshal . . . . .	3208
6.687activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	
Class Reference . . . . .	3209
6.687.1 Detailed Description . . . . .	3210
6.687.2 Constructor & Destructor Documentation . . . . .	3210
6.687.2.1 ReplayCommandMarshaller . . . . .	3210
6.687.2.2 ~ReplayCommandMarshaller . . . . .	3210
6.687.3 Member Function Documentation . . . . .	3210
6.687.3.1 createObject . . . . .	3210
6.687.3.2 getDataStructureType . . . . .	3210
6.687.3.3 looseMarshal . . . . .	3211
6.687.3.4 looseUnmarshal . . . . .	3211
6.687.3.5 tightMarshal1 . . . . .	3211
6.687.3.6 tightMarshal2 . . . . .	3212
6.687.3.7 tightUnmarshal . . . . .	3212
6.688activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller	
Class Reference . . . . .	3213

6.688.1 Detailed Description . . . . .	3214
6.688.2 Constructor & Destructor Documentation . . . . .	3214
6.688.2.1 ReplayCommandMarshaller . . . . .	3214
6.688.2.2 ~ReplayCommandMarshaller . . . . .	3214
6.688.3 Member Function Documentation . . . . .	3214
6.688.3.1 createObject . . . . .	3214
6.688.3.2 getDataStructureType . . . . .	3214
6.688.3.3 looseMarshal . . . . .	3215
6.688.3.4 looseUnmarshal . . . . .	3215
6.688.3.5 tightMarshal1 . . . . .	3215
6.688.3.6 tightMarshal2 . . . . .	3216
6.688.3.7 tightUnmarshal . . . . .	3216
6.689activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller	
Class Reference . . . . .	3217
6.689.1 Detailed Description . . . . .	3218
6.689.2 Constructor & Destructor Documentation . . . . .	3218
6.689.2.1 ReplayCommandMarshaller . . . . .	3218
6.689.2.2 ~ReplayCommandMarshaller . . . . .	3218
6.689.3 Member Function Documentation . . . . .	3218
6.689.3.1 createObject . . . . .	3218
6.689.3.2 getDataStructureType . . . . .	3218
6.689.3.3 looseMarshal . . . . .	3219
6.689.3.4 looseUnmarshal . . . . .	3219
6.689.3.5 tightMarshal1 . . . . .	3219
6.689.3.6 tightMarshal2 . . . . .	3220
6.689.3.7 tightUnmarshal . . . . .	3220
6.690activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Ref-	
erence . . . . .	3221
6.690.1 Constructor & Destructor Documentation . . . . .	3221
6.690.1.1 ResolveProducerExecutor . . . . .	3221
6.690.1.2 ~ResolveProducerExecutor . . . . .	3222
6.690.2 Member Function Documentation . . . . .	3222
6.690.2.1 getDestination . . . . .	3222
6.690.2.2 operator= . . . . .	3222

6.691 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference . . . . .	3222
6.691.1 Constructor & Destructor Documentation . . . . .	3222
6.691.1.1 ResolveReceiveExecutor . . . . .	3222
6.691.1.2 ~ResolveReceiveExecutor . . . . .	3223
6.691.2 Member Function Documentation . . . . .	3223
6.691.2.1 getDestination . . . . .	3223
6.691.2.2 operator= . . . . .	3223
6.692 decaf::internal::util::Resource Class Reference . . . . .	3223
6.692.1 Detailed Description . . . . .	3223
6.692.2 Constructor & Destructor Documentation . . . . .	3223
6.692.2.1 ~Resource . . . . .	3223
6.693 decaf::internal::util::ResourceLifecycleManager Class Reference . . . . .	3224
6.693.1 Detailed Description . . . . .	3224
6.693.2 Constructor & Destructor Documentation . . . . .	3224
6.693.2.1 ResourceLifecycleManager . . . . .	3224
6.693.2.2 ~ResourceLifecycleManager . . . . .	3224
6.693.3 Member Function Documentation . . . . .	3224
6.693.3.1 addResource . . . . .	3224
6.693.3.2 destroyResources . . . . .	3224
6.694 activemq::cmsutil::ResourceLifecycleManager Class Reference . . . . .	3224
6.694.1 Detailed Description . . . . .	3225
6.694.2 Constructor & Destructor Documentation . . . . .	3225
6.694.2.1 ResourceLifecycleManager . . . . .	3225
6.694.2.2 ResourceLifecycleManager . . . . .	3225
6.694.2.3 ~ResourceLifecycleManager . . . . .	3226
6.694.3 Member Function Documentation . . . . .	3226
6.694.3.1 addConnection . . . . .	3226
6.694.3.2 addDestination . . . . .	3226
6.694.3.3 addMessageConsumer . . . . .	3226
6.694.3.4 addMessageProducer . . . . .	3226
6.694.3.5 addSession . . . . .	3227
6.694.3.6 destroy . . . . .	3227
6.694.3.7 operator= . . . . .	3227

6.694.3.8	releaseAll	3227
6.695	activemq::commands::Response Class Reference	3227
6.695.1	Constructor & Destructor Documentation	3228
6.695.1.1	Response	3228
6.695.1.2	~Response	3228
6.695.2	Member Function Documentation	3228
6.695.2.1	cloneDataStructure	3228
6.695.2.2	copyDataStructure	3229
6.695.2.3	equals	3229
6.695.2.4	getCorrelationId	3229
6.695.2.5	getDataStructureType	3230
6.695.2.6	isResponse	3230
6.695.2.7	setCorrelationId	3230
6.695.2.8	toString	3230
6.695.2.9	visit	3230
6.695.3	Field Documentation	3231
6.695.3.1	correlationId	3231
6.695.3.2	ID_RESPONSE	3231
6.696	activemq::transport::mock::ResponseBuilder Class Reference	3231
6.696.1	Detailed Description	3231
6.696.2	Constructor & Destructor Documentation	3232
6.696.2.1	~ResponseBuilder	3232
6.696.3	Member Function Documentation	3232
6.696.3.1	buildIncomingCommands	3232
6.696.3.2	buildResponse	3232
6.697	activemq::transport::correlator::ResponseCorrelator Class Reference	3232
6.697.1	Detailed Description	3233
6.697.2	Constructor & Destructor Documentation	3233
6.697.2.1	ResponseCorrelator	3233
6.697.2.2	~ResponseCorrelator	3234
6.697.3	Member Function Documentation	3234
6.697.3.1	close	3234
6.697.3.2	onCommand	3234
6.697.3.3	oneway	3234



6.697.3.4 onTransportException . . . . .	3235
6.697.3.5 request . . . . .	3235
6.697.3.6 request . . . . .	3235
6.697.3.7 start . . . . .	3236
6.698activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class	
Reference . . . . .	3236
6.698.1 Detailed Description . . . . .	3237
6.698.2 Constructor & Destructor Documentation . . . . .	3237
6.698.2.1 ResponseMarshaller . . . . .	3237
6.698.2.2 ~ResponseMarshaller . . . . .	3237
6.698.3 Member Function Documentation . . . . .	3237
6.698.3.1 createObject . . . . .	3237
6.698.3.2 getDataStructureType . . . . .	3238
6.698.3.3 looseMarshal . . . . .	3238
6.698.3.4 looseUnmarshal . . . . .	3239
6.698.3.5 tightMarshal1 . . . . .	3239
6.698.3.6 tightMarshal2 . . . . .	3240
6.698.3.7 tightUnmarshal . . . . .	3240
6.699activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class	
Reference . . . . .	3241
6.699.1 Detailed Description . . . . .	3242
6.699.2 Constructor & Destructor Documentation . . . . .	3242
6.699.2.1 ResponseMarshaller . . . . .	3242
6.699.2.2 ~ResponseMarshaller . . . . .	3242
6.699.3 Member Function Documentation . . . . .	3242
6.699.3.1 createObject . . . . .	3242
6.699.3.2 getDataStructureType . . . . .	3242
6.699.3.3 looseMarshal . . . . .	3243
6.699.3.4 looseUnmarshal . . . . .	3243
6.699.3.5 tightMarshal1 . . . . .	3244
6.699.3.6 tightMarshal2 . . . . .	3244
6.699.3.7 tightUnmarshal . . . . .	3245
6.700activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class	
Reference . . . . .	3246
6.700.1 Detailed Description . . . . .	3246

6.700.2 Constructor & Destructor Documentation . . . . .	3247
6.700.2.1 ResponseMarshaller . . . . .	3247
6.700.2.2 ~ResponseMarshaller . . . . .	3247
6.700.3 Member Function Documentation . . . . .	3247
6.700.3.1 createObject . . . . .	3247
6.700.3.2 getDataStructureType . . . . .	3247
6.700.3.3 looseMarshal . . . . .	3247
6.700.3.4 looseUnmarshal . . . . .	3248
6.700.3.5 tightMarshal1 . . . . .	3248
6.700.3.6 tightMarshal2 . . . . .	3249
6.700.3.7 tightUnmarshal . . . . .	3250
6.701activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class	
Reference . . . . .	3250
6.701.1 Detailed Description . . . . .	3251
6.701.2 Constructor & Destructor Documentation . . . . .	3251
6.701.2.1 ResponseMarshaller . . . . .	3251
6.701.2.2 ~ResponseMarshaller . . . . .	3251
6.701.3 Member Function Documentation . . . . .	3251
6.701.3.1 createObject . . . . .	3251
6.701.3.2 getDataStructureType . . . . .	3252
6.701.3.3 looseMarshal . . . . .	3252
6.701.3.4 looseUnmarshal . . . . .	3253
6.701.3.5 tightMarshal1 . . . . .	3253
6.701.3.6 tightMarshal2 . . . . .	3254
6.701.3.7 tightUnmarshal . . . . .	3254
6.702activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class	
Reference . . . . .	3255
6.702.1 Detailed Description . . . . .	3256
6.702.2 Constructor & Destructor Documentation . . . . .	3256
6.702.2.1 ResponseMarshaller . . . . .	3256
6.702.2.2 ~ResponseMarshaller . . . . .	3256
6.702.3 Member Function Documentation . . . . .	3256
6.702.3.1 createObject . . . . .	3256
6.702.3.2 getDataStructureType . . . . .	3256

6.702.3.3 looseMarshal . . . . .	3257
6.702.3.4 looseUnmarshal . . . . .	3257
6.702.3.5 tightMarshal1 . . . . .	3258
6.702.3.6 tightMarshal2 . . . . .	3258
6.702.3.7 tightUnmarshal . . . . .	3259
6.703activemq::wireformat::openwire::marshal::v6::ResponseMarshaller Class Reference . . . . .	3260
6.703.1 Detailed Description . . . . .	3260
6.703.2 Constructor & Destructor Documentation . . . . .	3261
6.703.2.1 ResponseMarshaller . . . . .	3261
6.703.2.2 ~ResponseMarshaller . . . . .	3261
6.703.3 Member Function Documentation . . . . .	3261
6.703.3.1 createObject . . . . .	3261
6.703.3.2 getDataStructureType . . . . .	3261
6.703.3.3 looseMarshal . . . . .	3261
6.703.3.4 looseUnmarshal . . . . .	3262
6.703.3.5 tightMarshal1 . . . . .	3262
6.703.3.6 tightMarshal2 . . . . .	3263
6.703.3.7 tightUnmarshal . . . . .	3264
6.704decaf::lang::Runnable Class Reference . . . . .	3264
6.704.1 Detailed Description . . . . .	3265
6.704.2 Constructor & Destructor Documentation . . . . .	3265
6.704.2.1 ~Runnable . . . . .	3265
6.704.3 Member Function Documentation . . . . .	3265
6.704.3.1 run . . . . .	3265
6.705decaf::lang::Runtime Class Reference . . . . .	3265
6.705.1 Constructor & Destructor Documentation . . . . .	3266
6.705.1.1 ~Runtime . . . . .	3266
6.705.2 Member Function Documentation . . . . .	3266
6.705.2.1 getRuntime . . . . .	3266
6.705.2.2 initializeRuntime . . . . .	3266
6.705.2.3 initializeRuntime . . . . .	3266
6.705.2.4 shutdownRuntime . . . . .	3267
6.706decaf::lang::exceptions::RuntimeException Class Reference . . . . .	3267

6.706.1 Constructor & Destructor Documentation . . . . .	3268
6.706.1.1 RuntimeException . . . . .	3268
6.706.1.2 RuntimeException . . . . .	3268
6.706.1.3 RuntimeException . . . . .	3268
6.706.1.4 RuntimeException . . . . .	3268
6.706.1.5 RuntimeException . . . . .	3268
6.706.1.6 RuntimeException . . . . .	3269
6.706.1.7 ~RuntimeException . . . . .	3269
6.706.2 Member Function Documentation . . . . .	3269
6.706.2.1 clone . . . . .	3269
6.707decaf::security::SecureRandom Class Reference . . . . .	3269
6.707.1 Detailed Description . . . . .	3271
6.707.2 Constructor & Destructor Documentation . . . . .	3271
6.707.2.1 SecureRandom . . . . .	3271
6.707.2.2 SecureRandom . . . . .	3272
6.707.2.3 SecureRandom . . . . .	3272
6.707.2.4 ~SecureRandom . . . . .	3272
6.707.3 Member Function Documentation . . . . .	3272
6.707.3.1 next . . . . .	3272
6.707.3.2 nextBytes . . . . .	3273
6.707.3.3 nextBytes . . . . .	3273
6.707.3.4 setSeed . . . . .	3274
6.707.3.5 setSeed . . . . .	3274
6.707.3.6 setSeed . . . . .	3274
6.708decaf::internal::security::SecureRandomImpl Class Reference . . . . .	3275
6.708.1 Detailed Description . . . . .	3275
6.708.2 Constructor & Destructor Documentation . . . . .	3276
6.708.2.1 SecureRandomImpl . . . . .	3276
6.708.2.2 ~SecureRandomImpl . . . . .	3276
6.708.2.3 SecureRandomImpl . . . . .	3276
6.708.2.4 ~SecureRandomImpl . . . . .	3276
6.708.3 Member Function Documentation . . . . .	3276
6.708.3.1 providerGenerateSeed . . . . .	3276
6.708.3.2 providerGenerateSeed . . . . .	3276

6.708.3.3 providerNextBytes . . . . .	3277
6.708.3.4 providerNextBytes . . . . .	3277
6.708.3.5 providerSetSeed . . . . .	3277
6.708.3.6 providerSetSeed . . . . .	3278
6.709decaf::security::SecureRandomSpi Class Reference . . . . .	3278
6.709.1 Detailed Description . . . . .	3278
6.709.2 Constructor & Destructor Documentation . . . . .	3279
6.709.2.1 SecureRandomSpi . . . . .	3279
6.709.2.2 ~SecureRandomSpi . . . . .	3279
6.709.3 Member Function Documentation . . . . .	3279
6.709.3.1 providerGenerateSeed . . . . .	3279
6.709.3.2 providerNextBytes . . . . .	3279
6.709.3.3 providerSetSeed . . . . .	3279
6.710decaf::util::concurrent::Semaphore Class Reference . . . . .	3280
6.710.1 Detailed Description . . . . .	3281
6.710.2 Constructor & Destructor Documentation . . . . .	3283
6.710.2.1 Semaphore . . . . .	3283
6.710.2.2 Semaphore . . . . .	3283
6.710.2.3 ~Semaphore . . . . .	3283
6.710.3 Member Function Documentation . . . . .	3283
6.710.3.1 acquire . . . . .	3283
6.710.3.2 acquire . . . . .	3284
6.710.3.3 acquireUninterruptibly . . . . .	3285
6.710.3.4 acquireUninterruptibly . . . . .	3285
6.710.3.5 availablePermits . . . . .	3286
6.710.3.6 drainPermits . . . . .	3286
6.710.3.7 isFair . . . . .	3286
6.710.3.8 release . . . . .	3286
6.710.3.9 release . . . . .	3287
6.710.3.10 toString . . . . .	3287
6.710.3.11 tryAcquire . . . . .	3287
6.710.3.12 tryAcquire . . . . .	3288
6.710.3.13 tryAcquire . . . . .	3289
6.710.3.14 tryAcquire . . . . .	3290

6.711activemq::cmsutil::CmsTemplate::SendExecutor Class Reference . . . .	3290
6.711.1 Constructor & Destructor Documentation . . . . .	3291
6.711.1.1 SendExecutor . . . . .	3291
6.711.1.2 SendExecutor . . . . .	3291
6.711.1.3 ~SendExecutor . . . . .	3291
6.711.2 Member Function Documentation . . . . .	3291
6.711.2.1 doInCms . . . . .	3291
6.711.2.2 operator= . . . . .	3292
6.712decaf::net::ServerSocket Class Reference . . . . .	3292
6.712.1 Detailed Description . . . . .	3294
6.712.2 Constructor & Destructor Documentation . . . . .	3294
6.712.2.1 ServerSocket . . . . .	3294
6.712.2.2 ServerSocket . . . . .	3294
6.712.2.3 ServerSocket . . . . .	3294
6.712.2.4 ServerSocket . . . . .	3295
6.712.2.5 ~ServerSocket . . . . .	3295
6.712.2.6 ServerSocket . . . . .	3296
6.712.3 Member Function Documentation . . . . .	3296
6.712.3.1 accept . . . . .	3296
6.712.3.2 bind . . . . .	3296
6.712.3.3 bind . . . . .	3297
6.712.3.4 checkClosed . . . . .	3297
6.712.3.5 close . . . . .	3297
6.712.3.6 ensureCreated . . . . .	3298
6.712.3.7 getDefaultBacklog . . . . .	3298
6.712.3.8 getLocalPort . . . . .	3298
6.712.3.9 getReceiveBufferSize . . . . .	3298
6.712.3.10getReuseAddress . . . . .	3298
6.712.3.11getSoTimeout . . . . .	3299
6.712.3.12mplAccept . . . . .	3299
6.712.3.13sBound . . . . .	3299
6.712.3.14sClosed . . . . .	3299
6.712.3.15setReceiveBufferSize . . . . .	3299
6.712.3.16setReuseAddress . . . . .	3300

6.712.3.17	setSocketImplFactory . . . . .	3300
6.712.3.18	setSoTimeout . . . . .	3301
6.712.3.19	setupSocketImpl . . . . .	3301
6.712.3.20	toString . . . . .	3301
6.713	decaf::net::ServerSocketFactory Class Reference . . . . .	3301
6.713.1	Detailed Description . . . . .	3302
6.713.2	Constructor & Destructor Documentation . . . . .	3302
6.713.2.1	ServerSocketFactory . . . . .	3302
6.713.2.2	~ServerSocketFactory . . . . .	3302
6.713.3	Member Function Documentation . . . . .	3302
6.713.3.1	createServerSocket . . . . .	3302
6.713.3.2	createServerSocket . . . . .	3303
6.713.3.3	createServerSocket . . . . .	3303
6.713.3.4	createServerSocket . . . . .	3304
6.713.3.5	getDefault . . . . .	3304
6.714	cms::Session Class Reference . . . . .	3305
6.714.1	Detailed Description . . . . .	3307
6.714.2	Member Enumeration Documentation . . . . .	3308
6.714.2.1	AcknowledgeMode . . . . .	3308
6.714.3	Constructor & Destructor Documentation . . . . .	3308
6.714.3.1	~Session . . . . .	3308
6.714.4	Member Function Documentation . . . . .	3309
6.714.4.1	close . . . . .	3309
6.714.4.2	commit . . . . .	3309
6.714.4.3	createBrowser . . . . .	3309
6.714.4.4	createBrowser . . . . .	3310
6.714.4.5	createBytesMessage . . . . .	3310
6.714.4.6	createBytesMessage . . . . .	3311
6.714.4.7	createConsumer . . . . .	3311
6.714.4.8	createConsumer . . . . .	3311
6.714.4.9	createConsumer . . . . .	3312
6.714.4.10	createDurableConsumer . . . . .	3313
6.714.4.11	createMapMessage . . . . .	3314
6.714.4.12	createMessage . . . . .	3314

6.714.4.13createProducer . . . . .	3314
6.714.4.14createQueue . . . . .	3315
6.714.4.15createStreamMessage . . . . .	3315
6.714.4.16createTemporaryQueue . . . . .	3315
6.714.4.17createTemporaryTopic . . . . .	3316
6.714.4.18createTextMessage . . . . .	3316
6.714.4.19createTextMessage . . . . .	3316
6.714.4.20createTopic . . . . .	3317
6.714.4.21getAcknowledgeMode . . . . .	3317
6.714.4.22sTransacted . . . . .	3317
6.714.4.23recover . . . . .	3318
6.714.4.24rollback . . . . .	3318
6.714.4.25unsubscribe . . . . .	3319
6.715activemq::cmsutil::SessionCallback Class Reference . . . . .	3319
6.715.1 Detailed Description . . . . .	3319
6.715.2 Constructor & Destructor Documentation . . . . .	3320
6.715.2.1 ~SessionCallback . . . . .	3320
6.715.3 Member Function Documentation . . . . .	3320
6.715.3.1 doInCms . . . . .	3320
6.716activemq::commands::SessionId Class Reference . . . . .	3320
6.716.1 Member Typedef Documentation . . . . .	3321
6.716.1.1 COMPARATOR . . . . .	3321
6.716.2 Constructor & Destructor Documentation . . . . .	3321
6.716.2.1 SessionId . . . . .	3322
6.716.2.2 SessionId . . . . .	3322
6.716.2.3 SessionId . . . . .	3322
6.716.2.4 SessionId . . . . .	3322
6.716.2.5 SessionId . . . . .	3322
6.716.2.6 ~SessionId . . . . .	3322
6.716.3 Member Function Documentation . . . . .	3322
6.716.3.1 cloneDataStructure . . . . .	3322
6.716.3.2 compareTo . . . . .	3322
6.716.3.3 copyDataStructure . . . . .	3322
6.716.3.4 equals . . . . .	3322



6.716.3.5 equals . . . . .	3323
6.716.3.6 getConnectionId . . . . .	3323
6.716.3.7 getConnectionId . . . . .	3323
6.716.3.8 getDataStructureType . . . . .	3323
6.716.3.9 getParentId . . . . .	3323
6.716.3.10getValue . . . . .	3323
6.716.3.11operator< . . . . .	3323
6.716.3.12operator= . . . . .	3323
6.716.3.13operator== . . . . .	3323
6.716.3.14setConnectionId . . . . .	3323
6.716.3.15setValue . . . . .	3324
6.716.3.16toString . . . . .	3324
6.716.4 Field Documentation . . . . .	3324
6.716.4.1 connectionId . . . . .	3324
6.716.4.2 ID_SESSIONID . . . . .	3324
6.716.4.3 value . . . . .	3324
6.717activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class	
Reference . . . . .	3324
6.717.1 Detailed Description . . . . .	3325
6.717.2 Constructor & Destructor Documentation . . . . .	3325
6.717.2.1 SessionIdMarshaller . . . . .	3325
6.717.2.2 ~SessionIdMarshaller . . . . .	3325
6.717.3 Member Function Documentation . . . . .	3325
6.717.3.1 createObject . . . . .	3325
6.717.3.2 getDataStructureType . . . . .	3326
6.717.3.3 looseMarshal . . . . .	3326
6.717.3.4 looseUnmarshal . . . . .	3326
6.717.3.5 tightMarshal1 . . . . .	3327
6.717.3.6 tightMarshal2 . . . . .	3327
6.717.3.7 tightUnmarshal . . . . .	3328
6.718activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class	
Reference . . . . .	3328
6.718.1 Detailed Description . . . . .	3329
6.718.2 Constructor & Destructor Documentation . . . . .	3329

6.718.2.1 SessionIdMarshaller . . . . .	3329
6.718.2.2 ~SessionIdMarshaller . . . . .	3329
6.718.3 Member Function Documentation . . . . .	3329
6.718.3.1 createObject . . . . .	3329
6.718.3.2 getDataStructureType . . . . .	3330
6.718.3.3 looseMarshal . . . . .	3330
6.718.3.4 looseUnmarshal . . . . .	3330
6.718.3.5 tightMarshal1 . . . . .	3331
6.718.3.6 tightMarshal2 . . . . .	3331
6.718.3.7 tightUnmarshal . . . . .	3332
6.719activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class	
Reference . . . . .	3332
6.719.1 Detailed Description . . . . .	3333
6.719.2 Constructor & Destructor Documentation . . . . .	3333
6.719.2.1 SessionIdMarshaller . . . . .	3333
6.719.2.2 ~SessionIdMarshaller . . . . .	3333
6.719.3 Member Function Documentation . . . . .	3333
6.719.3.1 createObject . . . . .	3333
6.719.3.2 getDataStructureType . . . . .	3334
6.719.3.3 looseMarshal . . . . .	3334
6.719.3.4 looseUnmarshal . . . . .	3334
6.719.3.5 tightMarshal1 . . . . .	3335
6.719.3.6 tightMarshal2 . . . . .	3335
6.719.3.7 tightUnmarshal . . . . .	3336
6.720activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class	
Reference . . . . .	3336
6.720.1 Detailed Description . . . . .	3337
6.720.2 Constructor & Destructor Documentation . . . . .	3337
6.720.2.1 SessionIdMarshaller . . . . .	3337
6.720.2.2 ~SessionIdMarshaller . . . . .	3337
6.720.3 Member Function Documentation . . . . .	3337
6.720.3.1 createObject . . . . .	3337
6.720.3.2 getDataStructureType . . . . .	3338
6.720.3.3 looseMarshal . . . . .	3338

6.720.3.4 looseUnmarshal . . . . .	3338
6.720.3.5 tightMarshal1 . . . . .	3339
6.720.3.6 tightMarshal2 . . . . .	3339
6.720.3.7 tightUnmarshal . . . . .	3340
6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class	
Reference . . . . .	3340
6.721.1 Detailed Description . . . . .	3341
6.721.2 Constructor & Destructor Documentation . . . . .	3341
6.721.2.1 SessionIdMarshaller . . . . .	3341
6.721.2.2 ~SessionIdMarshaller . . . . .	3341
6.721.3 Member Function Documentation . . . . .	3341
6.721.3.1 createObject . . . . .	3341
6.721.3.2 getDataStructureType . . . . .	3342
6.721.3.3 looseMarshal . . . . .	3342
6.721.3.4 looseUnmarshal . . . . .	3342
6.721.3.5 tightMarshal1 . . . . .	3343
6.721.3.6 tightMarshal2 . . . . .	3343
6.721.3.7 tightUnmarshal . . . . .	3344
6.722 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class	
Reference . . . . .	3344
6.722.1 Detailed Description . . . . .	3345
6.722.2 Constructor & Destructor Documentation . . . . .	3345
6.722.2.1 SessionIdMarshaller . . . . .	3345
6.722.2.2 ~SessionIdMarshaller . . . . .	3345
6.722.3 Member Function Documentation . . . . .	3345
6.722.3.1 createObject . . . . .	3345
6.722.3.2 getDataStructureType . . . . .	3346
6.722.3.3 looseMarshal . . . . .	3346
6.722.3.4 looseUnmarshal . . . . .	3346
6.722.3.5 tightMarshal1 . . . . .	3347
6.722.3.6 tightMarshal2 . . . . .	3347
6.722.3.7 tightUnmarshal . . . . .	3348
6.723 activemq::commands::SessionInfo Class Reference . . . . .	3348
6.723.1 Constructor & Destructor Documentation . . . . .	3349

6.723.1.1 SessionInfo . . . . .	3349
6.723.1.2 ~SessionInfo . . . . .	3349
6.723.2 Member Function Documentation . . . . .	3349
6.723.2.1 cloneDataStructure . . . . .	3349
6.723.2.2 copyDataStructure . . . . .	3350
6.723.2.3 createRemoveCommand . . . . .	3350
6.723.2.4 equals . . . . .	3350
6.723.2.5 getAckMode . . . . .	3350
6.723.2.6 getDataStructureType . . . . .	3350
6.723.2.7 getSessionId . . . . .	3350
6.723.2.8 getSessionId . . . . .	3351
6.723.2.9 setAckMode . . . . .	3351
6.723.2.10setSessionId . . . . .	3351
6.723.2.11toString . . . . .	3351
6.723.2.12visit . . . . .	3351
6.723.3 Field Documentation . . . . .	3351
6.723.3.1 ID_SESSIONINFO . . . . .	3351
6.723.3.2 sessionId . . . . .	3351
6.724activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class	
Reference . . . . .	3352
6.724.1 Detailed Description . . . . .	3352
6.724.2 Constructor & Destructor Documentation . . . . .	3353
6.724.2.1 SessionInfoMarshaller . . . . .	3353
6.724.2.2 ~SessionInfoMarshaller . . . . .	3353
6.724.3 Member Function Documentation . . . . .	3353
6.724.3.1 createObject . . . . .	3353
6.724.3.2 getDataStructureType . . . . .	3353
6.724.3.3 looseMarshal . . . . .	3353
6.724.3.4 looseUnmarshal . . . . .	3354
6.724.3.5 tightMarshal1 . . . . .	3354
6.724.3.6 tightMarshal2 . . . . .	3355
6.724.3.7 tightUnmarshal . . . . .	3355
6.725activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class	
Reference . . . . .	3356

6.725.1 Detailed Description . . . . .	3356
6.725.2 Constructor & Destructor Documentation . . . . .	3357
6.725.2.1 SessionInfoMarshaller . . . . .	3357
6.725.2.2 ~SessionInfoMarshaller . . . . .	3357
6.725.3 Member Function Documentation . . . . .	3357
6.725.3.1 createObject . . . . .	3357
6.725.3.2 getDataStructureType . . . . .	3357
6.725.3.3 looseMarshal . . . . .	3357
6.725.3.4 looseUnmarshal . . . . .	3358
6.725.3.5 tightMarshal1 . . . . .	3358
6.725.3.6 tightMarshal2 . . . . .	3359
6.725.3.7 tightUnmarshal . . . . .	3359
6.726activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class	
Reference . . . . .	3360
6.726.1 Detailed Description . . . . .	3360
6.726.2 Constructor & Destructor Documentation . . . . .	3361
6.726.2.1 SessionInfoMarshaller . . . . .	3361
6.726.2.2 ~SessionInfoMarshaller . . . . .	3361
6.726.3 Member Function Documentation . . . . .	3361
6.726.3.1 createObject . . . . .	3361
6.726.3.2 getDataStructureType . . . . .	3361
6.726.3.3 looseMarshal . . . . .	3361
6.726.3.4 looseUnmarshal . . . . .	3362
6.726.3.5 tightMarshal1 . . . . .	3362
6.726.3.6 tightMarshal2 . . . . .	3363
6.726.3.7 tightUnmarshal . . . . .	3363
6.727activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class	
Reference . . . . .	3364
6.727.1 Detailed Description . . . . .	3364
6.727.2 Constructor & Destructor Documentation . . . . .	3365
6.727.2.1 SessionInfoMarshaller . . . . .	3365
6.727.2.2 ~SessionInfoMarshaller . . . . .	3365
6.727.3 Member Function Documentation . . . . .	3365
6.727.3.1 createObject . . . . .	3365

6.727.3.2	getDataStructureType . . . . .	3365
6.727.3.3	looseMarshal . . . . .	3365
6.727.3.4	looseUnmarshal . . . . .	3366
6.727.3.5	tightMarshal1 . . . . .	3366
6.727.3.6	tightMarshal2 . . . . .	3367
6.727.3.7	tightUnmarshal . . . . .	3367
6.728	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class	
	Reference . . . . .	3368
6.728.1	Detailed Description . . . . .	3368
6.728.2	Constructor & Destructor Documentation . . . . .	3369
	6.728.2.1 SessionInfoMarshaller . . . . .	3369
	6.728.2.2 ~SessionInfoMarshaller . . . . .	3369
6.728.3	Member Function Documentation . . . . .	3369
	6.728.3.1 createObject . . . . .	3369
	6.728.3.2 getDataStructureType . . . . .	3369
	6.728.3.3 looseMarshal . . . . .	3369
	6.728.3.4 looseUnmarshal . . . . .	3370
	6.728.3.5 tightMarshal1 . . . . .	3370
	6.728.3.6 tightMarshal2 . . . . .	3371
	6.728.3.7 tightUnmarshal . . . . .	3371
6.729	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class	
	Reference . . . . .	3372
6.729.1	Detailed Description . . . . .	3372
6.729.2	Constructor & Destructor Documentation . . . . .	3373
	6.729.2.1 SessionInfoMarshaller . . . . .	3373
	6.729.2.2 ~SessionInfoMarshaller . . . . .	3373
6.729.3	Member Function Documentation . . . . .	3373
	6.729.3.1 createObject . . . . .	3373
	6.729.3.2 getDataStructureType . . . . .	3373
	6.729.3.3 looseMarshal . . . . .	3373
	6.729.3.4 looseUnmarshal . . . . .	3374
	6.729.3.5 tightMarshal1 . . . . .	3374
	6.729.3.6 tightMarshal2 . . . . .	3375
	6.729.3.7 tightUnmarshal . . . . .	3375

6.730	activemq::cmsutil::SessionPool Class Reference . . . . .	3376
6.730.1	Detailed Description . . . . .	3376
6.730.2	Constructor & Destructor Documentation . . . . .	3376
6.730.2.1	SessionPool . . . . .	3376
6.730.2.2	SessionPool . . . . .	3376
6.730.2.3	~SessionPool . . . . .	3377
6.730.3	Member Function Documentation . . . . .	3377
6.730.3.1	getResourceLifecycleManager . . . . .	3377
6.730.3.2	operator= . . . . .	3377
6.730.3.3	returnSession . . . . .	3377
6.730.3.4	takeSession . . . . .	3377
6.731	activemq::state::SessionState Class Reference . . . . .	3378
6.731.1	Constructor & Destructor Documentation . . . . .	3378
6.731.1.1	SessionState . . . . .	3378
6.731.1.2	~SessionState . . . . .	3378
6.731.2	Member Function Documentation . . . . .	3378
6.731.2.1	addConsumer . . . . .	3378
6.731.2.2	addProducer . . . . .	3378
6.731.2.3	checkShutdown . . . . .	3379
6.731.2.4	getConsumerState . . . . .	3379
6.731.2.5	getConsumerStates . . . . .	3379
6.731.2.6	getInfo . . . . .	3379
6.731.2.7	getProducerState . . . . .	3379
6.731.2.8	getProducerStates . . . . .	3379
6.731.2.9	removeConsumer . . . . .	3379
6.731.2.10	removeProducer . . . . .	3379
6.731.2.11	shutdown . . . . .	3379
6.731.2.12	toString . . . . .	3379
6.732	decaf::util::Set< E > Class Template Reference . . . . .	3379
6.732.1	Detailed Description . . . . .	3379
6.732.2	Constructor & Destructor Documentation . . . . .	3380
6.732.2.1	~Set . . . . .	3380
6.733	decaf::lang::Short Class Reference . . . . .	3380
6.733.1	Constructor & Destructor Documentation . . . . .	3382

6.733.1.1 Short . . . . .	3382
6.733.1.2 Short . . . . .	3382
6.733.1.3 ~Short . . . . .	3382
6.733.2 Member Function Documentation . . . . .	3382
6.733.2.1 byteValue . . . . .	3382
6.733.2.2 compareTo . . . . .	3383
6.733.2.3 compareTo . . . . .	3383
6.733.2.4 decode . . . . .	3383
6.733.2.5 doubleValue . . . . .	3384
6.733.2.6 equals . . . . .	3384
6.733.2.7 equals . . . . .	3384
6.733.2.8 floatValue . . . . .	3384
6.733.2.9 intValue . . . . .	3385
6.733.2.10longValue . . . . .	3385
6.733.2.11operator< . . . . .	3385
6.733.2.12operator< . . . . .	3385
6.733.2.13operator== . . . . .	3386
6.733.2.14operator== . . . . .	3386
6.733.2.15parseShort . . . . .	3386
6.733.2.16parseShort . . . . .	3387
6.733.2.17reverseBytes . . . . .	3387
6.733.2.18shortValue . . . . .	3388
6.733.2.19toString . . . . .	3388
6.733.2.20toString . . . . .	3388
6.733.2.21valueOf . . . . .	3388
6.733.2.22valueOf . . . . .	3388
6.733.2.23valueOf . . . . .	3389
6.733.3 Field Documentation . . . . .	3389
6.733.3.1 MAX_VALUE . . . . .	3389
6.733.3.2 MIN_VALUE . . . . .	3389
6.733.3.3 SIZE . . . . .	3390
6.734decaf::internal::nio::ShortArrayBuffer Class Reference . . . . .	3390
6.734.1 Constructor & Destructor Documentation . . . . .	3393
6.734.1.1 ShortArrayBuffer . . . . .	3394



6.734.1.2 ShortArrayBuffer . . . . .	3394
6.734.1.3 ShortArrayBuffer . . . . .	3394
6.734.1.4 ShortArrayBuffer . . . . .	3395
6.734.1.5 ~ShortArrayBuffer . . . . .	3395
6.734.2 Member Function Documentation . . . . .	3395
6.734.2.1 array . . . . .	3395
6.734.2.2 arrayOffset . . . . .	3396
6.734.2.3 asReadOnlyBuffer . . . . .	3396
6.734.2.4 compact . . . . .	3397
6.734.2.5 duplicate . . . . .	3397
6.734.2.6 get . . . . .	3398
6.734.2.7 get . . . . .	3398
6.734.2.8 hasArray . . . . .	3398
6.734.2.9 isReadOnly . . . . .	3399
6.734.2.10put . . . . .	3399
6.734.2.11put . . . . .	3399
6.734.2.12setReadOnly . . . . .	3400
6.734.2.13slice . . . . .	3400
6.735decaf::nio::ShortBuffer Class Reference . . . . .	3401
6.735.1 Detailed Description . . . . .	3403
6.735.2 Constructor & Destructor Documentation . . . . .	3403
6.735.2.1 ShortBuffer . . . . .	3403
6.735.2.2 ~ShortBuffer . . . . .	3403
6.735.3 Member Function Documentation . . . . .	3403
6.735.3.1 allocate . . . . .	3403
6.735.3.2 array . . . . .	3404
6.735.3.3 arrayOffset . . . . .	3404
6.735.3.4 asReadOnlyBuffer . . . . .	3405
6.735.3.5 compact . . . . .	3405
6.735.3.6 compareTo . . . . .	3406
6.735.3.7 duplicate . . . . .	3406
6.735.3.8 equals . . . . .	3406
6.735.3.9 get . . . . .	3406
6.735.3.10get . . . . .	3406

6.735.3.11get . . . . .	3407
6.735.3.12get . . . . .	3408
6.735.3.13hasArray . . . . .	3408
6.735.3.14operator< . . . . .	3408
6.735.3.15operator== . . . . .	3408
6.735.3.16put . . . . .	3408
6.735.3.17put . . . . .	3409
6.735.3.18put . . . . .	3409
6.735.3.19put . . . . .	3410
6.735.3.20put . . . . .	3411
6.735.3.21slice . . . . .	3411
6.735.3.22toString . . . . .	3412
6.735.3.23wrap . . . . .	3412
6.735.3.24wrap . . . . .	3412
6.736activemq::commands::ShutdownInfo Class Reference . . . . .	3413
6.736.1 Constructor & Destructor Documentation . . . . .	3414
6.736.1.1 ShutdownInfo . . . . .	3414
6.736.1.2 ~ShutdownInfo . . . . .	3414
6.736.2 Member Function Documentation . . . . .	3414
6.736.2.1 cloneDataStructure . . . . .	3414
6.736.2.2 copyDataStructure . . . . .	3414
6.736.2.3 equals . . . . .	3414
6.736.2.4 getDataStructureType . . . . .	3415
6.736.2.5 isShutdownInfo . . . . .	3415
6.736.2.6 toString . . . . .	3415
6.736.2.7 visit . . . . .	3415
6.736.3 Field Documentation . . . . .	3416
6.736.3.1 ID_SHUTDOWNINFO . . . . .	3416
6.737activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller Class Reference . . . . .	3416
6.737.1 Detailed Description . . . . .	3417
6.737.2 Constructor & Destructor Documentation . . . . .	3417
6.737.2.1 ShutdownInfoMarshaller . . . . .	3417
6.737.2.2 ~ShutdownInfoMarshaller . . . . .	3417

6.737.3 Member Function Documentation . . . . .	3417
6.737.3.1 createObject . . . . .	3417
6.737.3.2 getDataStructureType . . . . .	3417
6.737.3.3 looseMarshal . . . . .	3418
6.737.3.4 looseUnmarshal . . . . .	3418
6.737.3.5 tightMarshal1 . . . . .	3418
6.737.3.6 tightMarshal2 . . . . .	3419
6.737.3.7 tightUnmarshal . . . . .	3419
6.738activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class	
Reference . . . . .	3420
6.738.1 Detailed Description . . . . .	3421
6.738.2 Constructor & Destructor Documentation . . . . .	3421
6.738.2.1 ShutdownInfoMarshaller . . . . .	3421
6.738.2.2 ~ShutdownInfoMarshaller . . . . .	3421
6.738.3 Member Function Documentation . . . . .	3421
6.738.3.1 createObject . . . . .	3421
6.738.3.2 getDataStructureType . . . . .	3421
6.738.3.3 looseMarshal . . . . .	3422
6.738.3.4 looseUnmarshal . . . . .	3422
6.738.3.5 tightMarshal1 . . . . .	3422
6.738.3.6 tightMarshal2 . . . . .	3423
6.738.3.7 tightUnmarshal . . . . .	3423
6.739activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class	
Reference . . . . .	3424
6.739.1 Detailed Description . . . . .	3425
6.739.2 Constructor & Destructor Documentation . . . . .	3425
6.739.2.1 ShutdownInfoMarshaller . . . . .	3425
6.739.2.2 ~ShutdownInfoMarshaller . . . . .	3425
6.739.3 Member Function Documentation . . . . .	3425
6.739.3.1 createObject . . . . .	3425
6.739.3.2 getDataStructureType . . . . .	3425
6.739.3.3 looseMarshal . . . . .	3426
6.739.3.4 looseUnmarshal . . . . .	3426
6.739.3.5 tightMarshal1 . . . . .	3426

6.739.3.6	tightMarshal2 . . . . .	3427
6.739.3.7	tightUnmarshal . . . . .	3427
6.740	activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference . . . . .	3428
6.740.1	Detailed Description . . . . .	3429
6.740.2	Constructor & Destructor Documentation . . . . .	3429
6.740.2.1	ShutdownInfoMarshaller . . . . .	3429
6.740.2.2	~ShutdownInfoMarshaller . . . . .	3429
6.740.3	Member Function Documentation . . . . .	3429
6.740.3.1	createObject . . . . .	3429
6.740.3.2	getDataStructureType . . . . .	3429
6.740.3.3	looseMarshal . . . . .	3430
6.740.3.4	looseUnmarshal . . . . .	3430
6.740.3.5	tightMarshal1 . . . . .	3430
6.740.3.6	tightMarshal2 . . . . .	3431
6.740.3.7	tightUnmarshal . . . . .	3431
6.741	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference . . . . .	3432
6.741.1	Detailed Description . . . . .	3433
6.741.2	Constructor & Destructor Documentation . . . . .	3433
6.741.2.1	ShutdownInfoMarshaller . . . . .	3433
6.741.2.2	~ShutdownInfoMarshaller . . . . .	3433
6.741.3	Member Function Documentation . . . . .	3433
6.741.3.1	createObject . . . . .	3433
6.741.3.2	getDataStructureType . . . . .	3433
6.741.3.3	looseMarshal . . . . .	3434
6.741.3.4	looseUnmarshal . . . . .	3434
6.741.3.5	tightMarshal1 . . . . .	3434
6.741.3.6	tightMarshal2 . . . . .	3435
6.741.3.7	tightUnmarshal . . . . .	3435
6.742	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference . . . . .	3436
6.742.1	Detailed Description . . . . .	3437
6.742.2	Constructor & Destructor Documentation . . . . .	3437
6.742.2.1	ShutdownInfoMarshaller . . . . .	3437

6.742.2.2 ~ShutdownInfoMarshaller . . . . .	3437
6.742.3 Member Function Documentation . . . . .	3437
6.742.3.1 createObject . . . . .	3437
6.742.3.2 getDataStructureType . . . . .	3437
6.742.3.3 looseMarshal . . . . .	3438
6.742.3.4 looseUnmarshal . . . . .	3438
6.742.3.5 tightMarshal1 . . . . .	3438
6.742.3.6 tightMarshal2 . . . . .	3439
6.742.3.7 tightUnmarshal . . . . .	3439
6.743decaf::security::SignatureException Class Reference . . . . .	3440
6.743.1 Constructor & Destructor Documentation . . . . .	3441
6.743.1.1 SignatureException . . . . .	3441
6.743.1.2 SignatureException . . . . .	3441
6.743.1.3 SignatureException . . . . .	3441
6.743.1.4 SignatureException . . . . .	3441
6.743.1.5 SignatureException . . . . .	3441
6.743.1.6 SignatureException . . . . .	3442
6.743.1.7 ~SignatureException . . . . .	3442
6.743.2 Member Function Documentation . . . . .	3442
6.743.2.1 clone . . . . .	3442
6.744decaf::util::logging::SimpleFormatter Class Reference . . . . .	3442
6.744.1 Detailed Description . . . . .	3443
6.744.2 Constructor & Destructor Documentation . . . . .	3443
6.744.2.1 SimpleFormatter . . . . .	3443
6.744.2.2 ~SimpleFormatter . . . . .	3443
6.744.3 Member Function Documentation . . . . .	3443
6.744.3.1 format . . . . .	3443
6.745decaf::util::logging::SimpleLogger Class Reference . . . . .	3444
6.745.1 Constructor & Destructor Documentation . . . . .	3444
6.745.1.1 SimpleLogger . . . . .	3444
6.745.1.2 ~SimpleLogger . . . . .	3444
6.745.2 Member Function Documentation . . . . .	3445
6.745.2.1 debug . . . . .	3445
6.745.2.2 error . . . . .	3445

6.745.2.3 fatal . . . . .	3445
6.745.2.4 info . . . . .	3445
6.745.2.5 log . . . . .	3445
6.745.2.6 mark . . . . .	3445
6.745.2.7 warn . . . . .	3445
6.746decaf::net::Socket Class Reference . . . . .	3445
6.746.1 Detailed Description . . . . .	3449
6.746.2 Constructor & Destructor Documentation . . . . .	3449
6.746.2.1 Socket . . . . .	3449
6.746.2.2 Socket . . . . .	3449
6.746.2.3 Socket . . . . .	3449
6.746.2.4 Socket . . . . .	3450
6.746.2.5 Socket . . . . .	3450
6.746.2.6 Socket . . . . .	3451
6.746.2.7 ~Socket . . . . .	3451
6.746.3 Member Function Documentation . . . . .	3451
6.746.3.1 accepted . . . . .	3451
6.746.3.2 bind . . . . .	3451
6.746.3.3 checkClosed . . . . .	3452
6.746.3.4 close . . . . .	3452
6.746.3.5 connect . . . . .	3452
6.746.3.6 connect . . . . .	3453
6.746.3.7 ensureCreated . . . . .	3453
6.746.3.8 getInetAddress . . . . .	3453
6.746.3.9 getInputStream . . . . .	3453
6.746.3.10getKeepAlive . . . . .	3454
6.746.3.11getLocalAddress . . . . .	3454
6.746.3.12getLocalPort . . . . .	3454
6.746.3.13getOOBInline . . . . .	3454
6.746.3.14getOutputStream . . . . .	3455
6.746.3.15getPort . . . . .	3455
6.746.3.16getReceiveBufferSize . . . . .	3455
6.746.3.17getReuseAddress . . . . .	3456
6.746.3.18getSendBufferSize . . . . .	3456

6.746.3.19getSoLinger . . . . .	3456
6.746.3.20getSoTimeout . . . . .	3457
6.746.3.21getTcpNoDelay . . . . .	3457
6.746.3.22getTrafficClass . . . . .	3457
6.746.3.23nitSocketImpl . . . . .	3458
6.746.3.24sBound . . . . .	3458
6.746.3.25sClosed . . . . .	3458
6.746.3.26sConnected . . . . .	3458
6.746.3.27sInputShutdown . . . . .	3458
6.746.3.28sOutputShutdown . . . . .	3458
6.746.3.29sendUrgentData . . . . .	3458
6.746.3.30setKeepAlive . . . . .	3459
6.746.3.31setOOBInline . . . . .	3459
6.746.3.32setReceiveBufferSize . . . . .	3459
6.746.3.33setReuseAddress . . . . .	3460
6.746.3.34setSendBufferSize . . . . .	3460
6.746.3.35setSocketImplFactory . . . . .	3460
6.746.3.36setSoLinger . . . . .	3461
6.746.3.37setSoTimeout . . . . .	3461
6.746.3.38setTcpNoDelay . . . . .	3462
6.746.3.39setTrafficClass . . . . .	3462
6.746.3.40shutdownInput . . . . .	3462
6.746.3.41shutdownOutput . . . . .	3463
6.746.3.42toString . . . . .	3463
6.746.4 Friends And Related Function Documentation . . . . .	3463
6.746.4.1 ServerSocket . . . . .	3463
6.746.5 Field Documentation . . . . .	3463
6.746.5.1 impl . . . . .	3463
6.747decaf::net::SocketAddress Class Reference . . . . .	3463
6.747.1 Detailed Description . . . . .	3464
6.747.2 Constructor & Destructor Documentation . . . . .	3464
6.747.2.1 ~SocketAddress . . . . .	3464
6.748decaf::net::SocketError Class Reference . . . . .	3464
6.748.1 Detailed Description . . . . .	3464

6.748.2 Member Function Documentation . . . . .	3464
6.748.2.1 getErrorCode . . . . .	3464
6.748.2.2 getErrorString . . . . .	3465
6.749decaf::net::SocketException Class Reference . . . . .	3465
6.749.1 Detailed Description . . . . .	3465
6.749.2 Constructor & Destructor Documentation . . . . .	3465
6.749.2.1 SocketException . . . . .	3465
6.749.2.2 SocketException . . . . .	3465
6.749.2.3 SocketException . . . . .	3466
6.749.2.4 SocketException . . . . .	3466
6.749.2.5 SocketException . . . . .	3466
6.749.2.6 SocketException . . . . .	3466
6.749.2.7 ~SocketException . . . . .	3466
6.749.3 Member Function Documentation . . . . .	3467
6.749.3.1 clone . . . . .	3467
6.750decaf::net::SocketFactory Class Reference . . . . .	3467
6.750.1 Detailed Description . . . . .	3468
6.750.2 Constructor & Destructor Documentation . . . . .	3468
6.750.2.1 SocketFactory . . . . .	3468
6.750.2.2 ~SocketFactory . . . . .	3468
6.750.3 Member Function Documentation . . . . .	3468
6.750.3.1 createSocket . . . . .	3468
6.750.3.2 createSocket . . . . .	3469
6.750.3.3 createSocket . . . . .	3469
6.750.3.4 createSocket . . . . .	3470
6.750.3.5 createSocket . . . . .	3470
6.750.3.6 getDefault . . . . .	3471
6.751decaf::internal::net::SocketFileDescriptor Class Reference . . . . .	3471
6.751.1 Detailed Description . . . . .	3472
6.751.2 Constructor & Destructor Documentation . . . . .	3472
6.751.2.1 SocketFileDescriptor . . . . .	3472
6.751.2.2 ~SocketFileDescriptor . . . . .	3472
6.751.3 Member Function Documentation . . . . .	3472
6.751.3.1 getValue . . . . .	3472



6.752decaf::net::SocketImpl Class Reference . . . . .	3472
6.752.1 Detailed Description . . . . .	3474
6.752.2 Constructor & Destructor Documentation . . . . .	3474
6.752.2.1 SocketImpl . . . . .	3474
6.752.2.2 ~SocketImpl . . . . .	3474
6.752.3 Member Function Documentation . . . . .	3474
6.752.3.1 accept . . . . .	3474
6.752.3.2 available . . . . .	3475
6.752.3.3 bind . . . . .	3475
6.752.3.4 close . . . . .	3475
6.752.3.5 connect . . . . .	3476
6.752.3.6 create . . . . .	3476
6.752.3.7 getFileDescriptor . . . . .	3477
6.752.3.8 getInetAddress . . . . .	3477
6.752.3.9 getInputStream . . . . .	3477
6.752.3.10getLocalAddress . . . . .	3477
6.752.3.11getLocalPort . . . . .	3478
6.752.3.12getOption . . . . .	3478
6.752.3.13getOutputStream . . . . .	3478
6.752.3.14getPort . . . . .	3478
6.752.3.15listen . . . . .	3479
6.752.3.16sendUrgentData . . . . .	3479
6.752.3.17setOption . . . . .	3479
6.752.3.18shutdownInput . . . . .	3480
6.752.3.19shutdownOutput . . . . .	3480
6.752.3.20supportsUrgentData . . . . .	3480
6.752.3.21toString . . . . .	3480
6.752.4 Field Documentation . . . . .	3480
6.752.4.1 address . . . . .	3481
6.752.4.2 fd . . . . .	3481
6.752.4.3 localPort . . . . .	3481
6.752.4.4 port . . . . .	3481
6.753decaf::net::SocketImplFactory Class Reference . . . . .	3481
6.753.1 Detailed Description . . . . .	3481

6.753.2 Constructor & Destructor Documentation . . . . .	3482
6.753.2.1 ~SocketImplFactory . . . . .	3482
6.753.3 Member Function Documentation . . . . .	3482
6.753.3.1 createSocketImpl . . . . .	3482
6.754decaf::net::SocketOptions Class Reference . . . . .	3482
6.754.1 Detailed Description . . . . .	3483
6.754.2 Constructor & Destructor Documentation . . . . .	3483
6.754.2.1 ~SocketOptions . . . . .	3483
6.754.3 Field Documentation . . . . .	3483
6.754.3.1 SOCKET_OPTION_BINDADDR . . . . .	3484
6.754.3.2 SOCKET_OPTION_BROADCAST . . . . .	3484
6.754.3.3 SOCKET_OPTION_IP_MULTICAST_IF . . . . .	3484
6.754.3.4 SOCKET_OPTION_IP_MULTICAST_IF2 . . . . .	3484
6.754.3.5 SOCKET_OPTION_IP_MULTICAST_LOOP . . . . .	3484
6.754.3.6 SOCKET_OPTION_IP_TOS . . . . .	3485
6.754.3.7 SOCKET_OPTION_KEEPAIVE . . . . .	3485
6.754.3.8 SOCKET_OPTION_LINGER . . . . .	3485
6.754.3.9 SOCKET_OPTION_OOBLINE . . . . .	3485
6.754.3.10SOCKET_OPTION_RCVBUF . . . . .	3486
6.754.3.11SOCKET_OPTION_REUSEADDR . . . . .	3486
6.754.3.12SOCKET_OPTION_SNDBUF . . . . .	3486
6.754.3.13SOCKET_OPTION_TCP_NODELAY . . . . .	3486
6.754.3.14SOCKET_OPTION_TIMEOUT . . . . .	3486
6.755decaf::net::SocketTimeoutException Class Reference . . . . .	3487
6.755.1 Constructor & Destructor Documentation . . . . .	3487
6.755.1.1 SocketTimeoutException . . . . .	3487
6.755.1.2 SocketTimeoutException . . . . .	3487
6.755.1.3 SocketTimeoutException . . . . .	3488
6.755.1.4 SocketTimeoutException . . . . .	3488
6.755.1.5 SocketTimeoutException . . . . .	3488
6.755.1.6 SocketTimeoutException . . . . .	3488
6.755.1.7 ~SocketTimeoutException . . . . .	3489
6.755.2 Member Function Documentation . . . . .	3489
6.755.2.1 clone . . . . .	3489

6.756decaf::net::ssl::SSLContext Class Reference . . . . .	3489
6.756.1 Detailed Description . . . . .	3490
6.756.2 Constructor & Destructor Documentation . . . . .	3490
6.756.2.1 SSLContext . . . . .	3490
6.756.2.2 ~SSLContext . . . . .	3490
6.756.3 Member Function Documentation . . . . .	3490
6.756.3.1 getDefault . . . . .	3490
6.756.3.2 getDefaultSSLParameters . . . . .	3490
6.756.3.3 getServerSocketFactory . . . . .	3491
6.756.3.4 getSocketFactory . . . . .	3491
6.756.3.5 getSupportedSSLParameters . . . . .	3491
6.756.3.6 setDefault . . . . .	3491
6.757decaf::net::ssl::SSLContextSpi Class Reference . . . . .	3492
6.757.1 Detailed Description . . . . .	3492
6.757.2 Constructor & Destructor Documentation . . . . .	3493
6.757.2.1 ~SSLContextSpi . . . . .	3493
6.757.3 Member Function Documentation . . . . .	3493
6.757.3.1 providerGetDefaultSSLParameters . . . . .	3493
6.757.3.2 providerGetServerSocketFactory . . . . .	3493
6.757.3.3 providerGetSocketFactory . . . . .	3494
6.757.3.4 providerGetSupportedSSLParameters . . . . .	3494
6.757.3.5 providerInit . . . . .	3494
6.758decaf::net::ssl::SSLParameters Class Reference . . . . .	3495
6.758.1 Constructor & Destructor Documentation . . . . .	3496
6.758.1.1 SSLParameters . . . . .	3496
6.758.1.2 SSLParameters . . . . .	3496
6.758.1.3 SSLParameters . . . . .	3496
6.758.1.4 ~SSLParameters . . . . .	3496
6.758.2 Member Function Documentation . . . . .	3496
6.758.2.1 getCipherSuites . . . . .	3496
6.758.2.2 getNeedClientAuth . . . . .	3497
6.758.2.3 getProtocols . . . . .	3497
6.758.2.4 getWantClientAuth . . . . .	3497
6.758.2.5 setCipherSuites . . . . .	3497

6.758.2.6 setNeedClientAuth . . . . .	3497
6.758.2.7 setProtocols . . . . .	3497
6.758.2.8 setWantClientAuth . . . . .	3498
6.759decaf::net::ssl::SSLServerSocket Class Reference . . . . .	3498
6.759.1 Detailed Description . . . . .	3499
6.759.2 Constructor & Destructor Documentation . . . . .	3499
6.759.2.1 SSLServerSocket . . . . .	3499
6.759.2.2 SSLServerSocket . . . . .	3500
6.759.2.3 SSLServerSocket . . . . .	3500
6.759.2.4 SSLServerSocket . . . . .	3501
6.759.2.5 ~SSLServerSocket . . . . .	3501
6.759.3 Member Function Documentation . . . . .	3501
6.759.3.1 getEnabledCipherSuites . . . . .	3501
6.759.3.2 getEnabledProtocols . . . . .	3501
6.759.3.3 getNeedClientAuth . . . . .	3502
6.759.3.4 getSupportedCipherSuites . . . . .	3502
6.759.3.5 getSupportedProtocols . . . . .	3502
6.759.3.6 getWantClientAuth . . . . .	3502
6.759.3.7 setEnabledCipherSuites . . . . .	3503
6.759.3.8 setEnabledProtocols . . . . .	3503
6.759.3.9 setNeedClientAuth . . . . .	3503
6.759.3.10setWantClientAuth . . . . .	3504
6.760decaf::net::ssl::SSLServerSocketFactory Class Reference . . . . .	3504
6.760.1 Detailed Description . . . . .	3505
6.760.2 Constructor & Destructor Documentation . . . . .	3505
6.760.2.1 SSLServerSocketFactory . . . . .	3505
6.760.2.2 ~SSLServerSocketFactory . . . . .	3505
6.760.3 Member Function Documentation . . . . .	3505
6.760.3.1 getDefault . . . . .	3505
6.760.3.2 getDefaultCipherSuites . . . . .	3505
6.760.3.3 getSupportedCipherSuites . . . . .	3506
6.761decaf::net::ssl::SSLSocket Class Reference . . . . .	3506
6.761.1 Detailed Description . . . . .	3508
6.761.2 Constructor & Destructor Documentation . . . . .	3508

6.761.2.1 SSLSocket . . . . .	3508
6.761.2.2 SSLSocket . . . . .	3508
6.761.2.3 SSLSocket . . . . .	3509
6.761.2.4 SSLSocket . . . . .	3509
6.761.2.5 SSLSocket . . . . .	3509
6.761.2.6 ~SSLSocket . . . . .	3510
6.761.3 Member Function Documentation . . . . .	3510
6.761.3.1 getEnabledCipherSuites . . . . .	3510
6.761.3.2 getEnabledProtocols . . . . .	3510
6.761.3.3 getNeedClientAuth . . . . .	3511
6.761.3.4 getSSLParameters . . . . .	3511
6.761.3.5 getSupportedCipherSuites . . . . .	3511
6.761.3.6 getSupportedProtocols . . . . .	3511
6.761.3.7 getUseClientMode . . . . .	3512
6.761.3.8 getWantClientAuth . . . . .	3512
6.761.3.9 setEnabledCipherSuites . . . . .	3512
6.761.3.10setEnabledProtocols . . . . .	3513
6.761.3.11setNeedClientAuth . . . . .	3513
6.761.3.12setSSLParameters . . . . .	3513
6.761.3.13setUseClientMode . . . . .	3514
6.761.3.14setWantClientAuth . . . . .	3514
6.761.3.15startHandshake . . . . .	3515
6.762decaf::net::ssl::SSLSocketFactory Class Reference . . . . .	3515
6.762.1 Detailed Description . . . . .	3516
6.762.2 Constructor & Destructor Documentation . . . . .	3516
6.762.2.1 SSLSocketFactory . . . . .	3516
6.762.2.2 ~SSLSocketFactory . . . . .	3516
6.762.3 Member Function Documentation . . . . .	3516
6.762.3.1 createSocket . . . . .	3516
6.762.3.2 getDefault . . . . .	3517
6.762.3.3 getDefaultCipherSuites . . . . .	3517
6.762.3.4 getSupportedCipherSuites . . . . .	3517
6.763activemq::transport::tcp::SslTransport Class Reference . . . . .	3518
6.763.1 Detailed Description . . . . .	3519

6.763.2 Constructor & Destructor Documentation . . . . .	3519
6.763.2.1 SslTransport . . . . .	3519
6.763.2.2 ~SslTransport . . . . .	3519
6.763.3 Member Function Documentation . . . . .	3519
6.763.3.1 configureSocket . . . . .	3519
6.763.3.2 createSocket . . . . .	3519
6.764activemq::transport::tcp::SslTransportFactory Class Reference . . . . .	3520
6.764.1 Constructor & Destructor Documentation . . . . .	3520
6.764.1.1 ~SslTransportFactory . . . . .	3520
6.764.2 Member Function Documentation . . . . .	3520
6.764.2.1 doCreateComposite . . . . .	3520
6.765activemq::commands::BrokerError::StackTraceElement Struct Reference	3521
6.765.1 Field Documentation . . . . .	3521
6.765.1.1 ClassName . . . . .	3521
6.765.1.2 FileName . . . . .	3521
6.765.1.3 LineNumber . . . . .	3521
6.765.1.4 MethodName . . . . .	3521
6.766decaf::internal::io::StandardErrorOutputStream Class Reference . . . . .	3521
6.766.1 Detailed Description . . . . .	3522
6.766.2 Constructor & Destructor Documentation . . . . .	3522
6.766.2.1 StandardErrorOutputStream . . . . .	3522
6.766.2.2 ~StandardErrorOutputStream . . . . .	3522
6.766.3 Member Function Documentation . . . . .	3523
6.766.3.1 close . . . . .	3523
6.766.3.2 doWriteArrayBounded . . . . .	3523
6.766.3.3 doWriteByte . . . . .	3523
6.766.3.4 flush . . . . .	3523
6.767decaf::internal::io::StandardInputStream Class Reference . . . . .	3524
6.767.1 Constructor & Destructor Documentation . . . . .	3524
6.767.1.1 StandardInputStream . . . . .	3524
6.767.1.2 ~StandardInputStream . . . . .	3524
6.767.2 Member Function Documentation . . . . .	3524
6.767.2.1 available . . . . .	3524
6.767.2.2 doReadByte . . . . .	3525

6.768	decaf::internal::io::StandardOutputStream Class Reference . . . . .	3525
6.768.1	Constructor & Destructor Documentation . . . . .	3526
6.768.1.1	StandardOutputStream . . . . .	3526
6.768.1.2	~StandardOutputStream . . . . .	3526
6.768.2	Member Function Documentation . . . . .	3526
6.768.2.1	close . . . . .	3526
6.768.2.2	doWriteArrayBounded . . . . .	3526
6.768.2.3	doWriteByte . . . . .	3526
6.768.2.4	flush . . . . .	3526
6.769	cms::Startable Class Reference . . . . .	3527
6.769.1	Detailed Description . . . . .	3527
6.769.2	Constructor & Destructor Documentation . . . . .	3527
6.769.2.1	~Startable . . . . .	3527
6.769.3	Member Function Documentation . . . . .	3527
6.769.3.1	start . . . . .	3527
6.770	decaf::lang::STATIC_CAST_TOKEN Struct Reference . . . . .	3528
6.771	activemq::core::ActiveMQConstants::StaticInitializer Class Reference . .	3528
6.771.1	Constructor & Destructor Documentation . . . . .	3528
6.771.1.1	StaticInitializer . . . . .	3528
6.771.1.2	~StaticInitializer . . . . .	3528
6.771.2	Field Documentation . . . . .	3529
6.771.2.1	destOptionMap . . . . .	3529
6.771.2.2	destOptions . . . . .	3529
6.771.2.3	uriParams . . . . .	3529
6.771.2.4	uriParamsMap . . . . .	3529
6.772	decaf::util::StlList< E > Class Template Reference . . . . .	3529
6.772.1	Detailed Description . . . . .	3534
6.772.2	Constructor & Destructor Documentation . . . . .	3534
6.772.2.1	StlList . . . . .	3534
6.772.2.2	StlList . . . . .	3534
6.772.2.3	StlList . . . . .	3535
6.772.2.4	~StlList . . . . .	3535
6.772.3	Member Function Documentation . . . . .	3535
6.772.3.1	add . . . . .	3535

6.772.3.2 add . . . . .	3536
6.772.3.3 addAll . . . . .	3536
6.772.3.4 clear . . . . .	3537
6.772.3.5 contains . . . . .	3537
6.772.3.6 copy . . . . .	3538
6.772.3.7 equals . . . . .	3538
6.772.3.8 get . . . . .	3538
6.772.3.9 indexOf . . . . .	3538
6.772.3.10isEmpty . . . . .	3539
6.772.3.11iterator . . . . .	3539
6.772.3.12iterator . . . . .	3539
6.772.3.13lastIndexOf . . . . .	3539
6.772.3.14listIterator . . . . .	3540
6.772.3.15listIterator . . . . .	3540
6.772.3.16listIterator . . . . .	3540
6.772.3.17listIterator . . . . .	3541
6.772.3.18remove . . . . .	3541
6.772.3.19remove . . . . .	3541
6.772.3.20set . . . . .	3542
6.772.3.21size . . . . .	3542
6.773decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference .	3543
6.773.1 Detailed Description . . . . .	3547
6.773.2 Constructor & Destructor Documentation . . . . .	3547
6.773.2.1 StlMap . . . . .	3547
6.773.2.2 StlMap . . . . .	3547
6.773.2.3 StlMap . . . . .	3547
6.773.2.4 ~StlMap . . . . .	3547
6.773.3 Member Function Documentation . . . . .	3547
6.773.3.1 clear . . . . .	3548
6.773.3.2 containsKey . . . . .	3548
6.773.3.3 containsValue . . . . .	3548
6.773.3.4 copy . . . . .	3549
6.773.3.5 copy . . . . .	3549
6.773.3.6 equals . . . . .	3549



6.773.3.7 equals . . . . .	3549
6.773.3.8 get . . . . .	3549
6.773.3.9 get . . . . .	3550
6.773.3.10isEmpty . . . . .	3550
6.773.3.11keySet . . . . .	3551
6.773.3.12lock . . . . .	3551
6.773.3.13notify . . . . .	3551
6.773.3.14notifyAll . . . . .	3552
6.773.3.15put . . . . .	3552
6.773.3.16putAll . . . . .	3552
6.773.3.17putAll . . . . .	3553
6.773.3.18remove . . . . .	3553
6.773.3.19size . . . . .	3554
6.773.3.20tryLock . . . . .	3554
6.773.3.21unlock . . . . .	3554
6.773.3.22values . . . . .	3554
6.773.3.23wait . . . . .	3555
6.773.3.24wait . . . . .	3555
6.773.3.25wait . . . . .	3556
6.774decaf::util::StlQueue< T > Class Template Reference . . . . .	3556
6.774.1 Detailed Description . . . . .	3558
6.774.2 Constructor & Destructor Documentation . . . . .	3558
6.774.2.1 StlQueue . . . . .	3558
6.774.2.2 ~StlQueue . . . . .	3558
6.774.3 Member Function Documentation . . . . .	3558
6.774.3.1 back . . . . .	3559
6.774.3.2 back . . . . .	3559
6.774.3.3 clear . . . . .	3559
6.774.3.4 empty . . . . .	3559
6.774.3.5 enqueueFront . . . . .	3559
6.774.3.6 front . . . . .	3559
6.774.3.7 front . . . . .	3560
6.774.3.8 getSafeValue . . . . .	3560
6.774.3.9 iterator . . . . .	3560

6.774.3.10lock . . . . .	3560
6.774.3.11notify . . . . .	3561
6.774.3.12notifyAll . . . . .	3561
6.774.3.13pop . . . . .	3561
6.774.3.14push . . . . .	3561
6.774.3.15reverse . . . . .	3562
6.774.3.16size . . . . .	3562
6.774.3.17toArray . . . . .	3562
6.774.3.18tryLock . . . . .	3562
6.774.3.19unlock . . . . .	3563
6.774.3.20wait . . . . .	3563
6.774.3.21wait . . . . .	3563
6.774.3.22wait . . . . .	3564
6.775decaf::util::StlSet< E > Class Template Reference . . . . .	3564
6.775.1 Detailed Description . . . . .	3567
6.775.2 Constructor & Destructor Documentation . . . . .	3567
6.775.2.1 StlSet . . . . .	3567
6.775.2.2 StlSet . . . . .	3567
6.775.2.3 StlSet . . . . .	3567
6.775.2.4 ~StlSet . . . . .	3567
6.775.3 Member Function Documentation . . . . .	3568
6.775.3.1 add . . . . .	3568
6.775.3.2 clear . . . . .	3568
6.775.3.3 contains . . . . .	3569
6.775.3.4 copy . . . . .	3569
6.775.3.5 equals . . . . .	3569
6.775.3.6 isEmpty . . . . .	3570
6.775.3.7 iterator . . . . .	3570
6.775.3.8 iterator . . . . .	3570
6.775.3.9 remove . . . . .	3570
6.775.3.10size . . . . .	3571
6.776activemq::wireformat::stomp::StompCommandConstants Class Reference	3571
6.776.1 Field Documentation . . . . .	3573
6.776.1.1 ABORT . . . . .	3573

6.776.1.2 ACK . . . . .	3573
6.776.1.3 ACK_AUTO . . . . .	3573
6.776.1.4 ACK_CLIENT . . . . .	3573
6.776.1.5 ACK_INDIVIDUAL . . . . .	3573
6.776.1.6 BEGIN . . . . .	3573
6.776.1.7 BYTES . . . . .	3573
6.776.1.8 COMMIT . . . . .	3573
6.776.1.9 CONNECT . . . . .	3573
6.776.1.10CONNECTED . . . . .	3573
6.776.1.11DISCONNECT . . . . .	3573
6.776.1.12ERROR_CMD . . . . .	3573
6.776.1.13HEADER_ACK . . . . .	3573
6.776.1.14HEADER_CLIENT_ID . . . . .	3573
6.776.1.15HEADER_CONSUMERPRIORITY . . . . .	3573
6.776.1.16HEADER_CONTENTLENGTH . . . . .	3574
6.776.1.17HEADER_CORRELATIONID . . . . .	3574
6.776.1.18HEADER_DESTINATION . . . . .	3574
6.776.1.19HEADER_DISPATCH_ASYNC . . . . .	3574
6.776.1.20HEADER_EXCLUSIVE . . . . .	3574
6.776.1.21HEADER_EXPIRES . . . . .	3574
6.776.1.22HEADER_ID . . . . .	3574
6.776.1.23HEADER_JMSPRIORITY . . . . .	3574
6.776.1.24HEADER_LOGIN . . . . .	3574
6.776.1.25HEADER_MAXPENDINGMSGLIMIT . . . . .	3574
6.776.1.26HEADER_MESSAGE . . . . .	3574
6.776.1.27HEADER_MESSAGEID . . . . .	3574
6.776.1.28HEADER_NOLOCAL . . . . .	3574
6.776.1.29HEADER_OLDSUBSCRIPTIONNAME . . . . .	3574
6.776.1.30HEADER_PASSWORD . . . . .	3574
6.776.1.31HEADER_PERSISTENT . . . . .	3574
6.776.1.32HEADER_PREFETCHSIZE . . . . .	3575
6.776.1.33HEADER_RECEIPT_REQUIRED . . . . .	3575
6.776.1.34HEADER_RECEIPTID . . . . .	3575
6.776.1.35HEADER_REDELIVERED . . . . .	3575

6.776.1.36	HEADER_REDELIVERYCOUNT . . . . .	3575
6.776.1.37	HEADER_REPLYTO . . . . .	3575
6.776.1.38	HEADER_REQUESTID . . . . .	3575
6.776.1.39	HEADER_RESPONSEID . . . . .	3575
6.776.1.40	HEADER_RETROACTIVE . . . . .	3575
6.776.1.41	HEADER_SELECTOR . . . . .	3575
6.776.1.42	HEADER_SESSIONID . . . . .	3575
6.776.1.43	HEADER_SUBSCRIPTION . . . . .	3575
6.776.1.44	HEADER_SUBSCRIPTIONNAME . . . . .	3575
6.776.1.45	HEADER_TIMESTAMP . . . . .	3575
6.776.1.46	HEADER_TRANSACTIONID . . . . .	3575
6.776.1.47	HEADER_TRANSFORMATION . . . . .	3575
6.776.1.48	HEADER_TRANSFORMATION_ERROR . . . . .	3576
6.776.1.49	HEADER_TYPE . . . . .	3576
6.776.1.50	MESSAGE . . . . .	3576
6.776.1.51	QUEUE_PREFIX . . . . .	3576
6.776.1.52	RECEIPT . . . . .	3576
6.776.1.53	SEND . . . . .	3576
6.776.1.54	SUBSCRIBE . . . . .	3576
6.776.1.55	TEMPQUEUE_PREFIX . . . . .	3576
6.776.1.56	TEMPTOPIC_PREFIX . . . . .	3576
6.776.1.57	TEXT . . . . .	3576
6.776.1.58	TOPIC_PREFIX . . . . .	3576
6.776.1.59	UNSUBSCRIBE . . . . .	3576
6.777	activemq::wireformat::stomp::StompFrame Class Reference . . . . .	3576
6.777.1	Detailed Description . . . . .	3578
6.777.2	Constructor & Destructor Documentation . . . . .	3578
6.777.2.1	StompFrame . . . . .	3578
6.777.2.2	~StompFrame . . . . .	3578
6.777.3	Member Function Documentation . . . . .	3578
6.777.3.1	clone . . . . .	3578
6.777.3.2	copy . . . . .	3578
6.777.3.3	fromStream . . . . .	3578
6.777.3.4	getBody . . . . .	3579

6.777.3.5	getBody . . . . .	3579
6.777.3.6	getBodyLength . . . . .	3579
6.777.3.7	getCommand . . . . .	3579
6.777.3.8	getProperties . . . . .	3579
6.777.3.9	getProperties . . . . .	3579
6.777.3.10	getProperty . . . . .	3579
6.777.3.11	hasProperty . . . . .	3580
6.777.3.12	removeProperty . . . . .	3580
6.777.3.13	setBody . . . . .	3580
6.777.3.14	setCommand . . . . .	3580
6.777.3.15	setProperty . . . . .	3581
6.777.3.16	toStream . . . . .	3581
6.778	activemq::wireformat::stomp::StompHelper Class Reference . . . . .	3581
6.778.1	Detailed Description . . . . .	3582
6.778.2	Constructor & Destructor Documentation . . . . .	3582
6.778.2.1	StompHelper . . . . .	3582
6.778.2.2	~StompHelper . . . . .	3582
6.778.3	Member Function Documentation . . . . .	3582
6.778.3.1	convertConsumerId . . . . .	3583
6.778.3.2	convertConsumerId . . . . .	3583
6.778.3.3	convertDestination . . . . .	3583
6.778.3.4	convertDestination . . . . .	3583
6.778.3.5	convertMessageId . . . . .	3584
6.778.3.6	convertMessageId . . . . .	3584
6.778.3.7	convertProducerId . . . . .	3584
6.778.3.8	convertProducerId . . . . .	3584
6.778.3.9	convertProperties . . . . .	3585
6.778.3.10	convertProperties . . . . .	3585
6.778.3.11	convertTransactionId . . . . .	3585
6.778.3.12	convertTransactionId . . . . .	3585
6.779	activemq::wireformat::stomp::StompWireFormat Class Reference . . . . .	3586
6.779.1	Constructor & Destructor Documentation . . . . .	3587
6.779.1.1	StompWireFormat . . . . .	3587
6.779.1.2	~StompWireFormat . . . . .	3587

6.779.2 Member Function Documentation . . . . .	3587
6.779.2.1 createNegotiator . . . . .	3587
6.779.2.2 getVersion . . . . .	3587
6.779.2.3 hasNegotiator . . . . .	3587
6.779.2.4 inReceive . . . . .	3588
6.779.2.5 marshal . . . . .	3588
6.779.2.6 setVersion . . . . .	3588
6.779.2.7 unmarshal . . . . .	3589
6.780activemq::wireformat::stomp::StompWireFormatFactory Class Reference	3589
6.780.1 Detailed Description . . . . .	3590
6.780.2 Constructor & Destructor Documentation . . . . .	3590
6.780.2.1 StompWireFormatFactory . . . . .	3590
6.780.2.2 ~StompWireFormatFactory . . . . .	3590
6.780.3 Member Function Documentation . . . . .	3590
6.780.3.1 createWireFormat . . . . .	3590
6.781cms::Stoppable Class Reference . . . . .	3590
6.781.1 Detailed Description . . . . .	3591
6.781.2 Constructor & Destructor Documentation . . . . .	3591
6.781.2.1 ~Stoppable . . . . .	3591
6.781.3 Member Function Documentation . . . . .	3591
6.781.3.1 stop . . . . .	3591
6.782decaf::util::logging::StreamHandler Class Reference . . . . .	3591
6.782.1 Detailed Description . . . . .	3592
6.782.2 Constructor & Destructor Documentation . . . . .	3593
6.782.2.1 StreamHandler . . . . .	3593
6.782.2.2 StreamHandler . . . . .	3593
6.782.2.3 ~StreamHandler . . . . .	3593
6.782.3 Member Function Documentation . . . . .	3593
6.782.3.1 close . . . . .	3593
6.782.3.2 close . . . . .	3593
6.782.3.3 flush . . . . .	3593
6.782.3.4 isLoggable . . . . .	3594
6.782.3.5 publish . . . . .	3594
6.782.3.6 setOutputStream . . . . .	3594

6.783cms::StreamMessage Class Reference . . . . .	3595
6.783.1 Detailed Description . . . . .	3597
6.783.2 Constructor & Destructor Documentation . . . . .	3597
6.783.2.1 ~StreamMessage . . . . .	3597
6.783.3 Member Function Documentation . . . . .	3597
6.783.3.1 readBoolean . . . . .	3597
6.783.3.2 readByte . . . . .	3598
6.783.3.3 readBytes . . . . .	3598
6.783.3.4 readBytes . . . . .	3599
6.783.3.5 readChar . . . . .	3600
6.783.3.6 readDouble . . . . .	3601
6.783.3.7 readFloat . . . . .	3601
6.783.3.8 readInt . . . . .	3602
6.783.3.9 readLong . . . . .	3602
6.783.3.10readShort . . . . .	3603
6.783.3.11readString . . . . .	3603
6.783.3.12readUnsignedShort . . . . .	3604
6.783.3.13writeBoolean . . . . .	3605
6.783.3.14writeByte . . . . .	3605
6.783.3.15writeBytes . . . . .	3605
6.783.3.16writeBytes . . . . .	3606
6.783.3.17writeChar . . . . .	3606
6.783.3.18writeDouble . . . . .	3607
6.783.3.19writeFloat . . . . .	3607
6.783.3.20writeInt . . . . .	3608
6.783.3.21writeLong . . . . .	3608
6.783.3.22writeShort . . . . .	3608
6.783.3.23writeString . . . . .	3609
6.783.3.24writeUnsignedShort . . . . .	3609
6.784decaf::lang::String Class Reference . . . . .	3610
6.784.1 Detailed Description . . . . .	3611
6.784.2 Constructor & Destructor Documentation . . . . .	3611
6.784.2.1 String . . . . .	3611
6.784.2.2 String . . . . .	3611

6.784.2.3 ~String . . . . .	3611
6.784.3 Member Function Documentation . . . . .	3611
6.784.3.1 charAt . . . . .	3611
6.784.3.2 isEmpty . . . . .	3612
6.784.3.3 length . . . . .	3612
6.784.3.4 subSequence . . . . .	3612
6.784.3.5 toString . . . . .	3613
6.785decaf::util::StringTokenizer Class Reference . . . . .	3613
6.785.1 Constructor & Destructor Documentation . . . . .	3613
6.785.1.1 StringTokenizer . . . . .	3614
6.785.1.2 ~StringTokenizer . . . . .	3614
6.785.2 Member Function Documentation . . . . .	3614
6.785.2.1 countTokens . . . . .	3614
6.785.2.2 hasMoreTokens . . . . .	3614
6.785.2.3 nextToken . . . . .	3614
6.785.2.4 nextToken . . . . .	3615
6.785.2.5 reset . . . . .	3615
6.785.2.6 toArray . . . . .	3616
6.786activemq::commands::SubscriptionInfo Class Reference . . . . .	3616
6.786.1 Constructor & Destructor Documentation . . . . .	3617
6.786.1.1 SubscriptionInfo . . . . .	3617
6.786.1.2 ~SubscriptionInfo . . . . .	3617
6.786.2 Member Function Documentation . . . . .	3617
6.786.2.1 cloneDataStructure . . . . .	3617
6.786.2.2 copyDataStructure . . . . .	3618
6.786.2.3 equals . . . . .	3618
6.786.2.4 getClientId . . . . .	3618
6.786.2.5 getClientId . . . . .	3618
6.786.2.6 getDataStructureType . . . . .	3618
6.786.2.7 getDestination . . . . .	3618
6.786.2.8 getDestination . . . . .	3618
6.786.2.9 getSelector . . . . .	3619
6.786.2.10getSelector . . . . .	3619
6.786.2.11getSubscriptionName . . . . .	3619



6.786.2.12	getSubscriptionName . . . . .	3619
6.786.2.13	getSubscribedDestination . . . . .	3619
6.786.2.14	getSubscribedDestination . . . . .	3619
6.786.2.15	setClientId . . . . .	3619
6.786.2.16	setDestination . . . . .	3619
6.786.2.17	setSelector . . . . .	3619
6.786.2.18	setSubscriptionName . . . . .	3619
6.786.2.19	setSubscribedDestination . . . . .	3619
6.786.2.20	toString . . . . .	3619
6.786.3	Field Documentation . . . . .	3620
6.786.3.1	clientId . . . . .	3620
6.786.3.2	destination . . . . .	3620
6.786.3.3	ID_SUBSCRIPTIONINFO . . . . .	3620
6.786.3.4	selector . . . . .	3620
6.786.3.5	subscriptionName . . . . .	3620
6.786.3.6	subscribedDestination . . . . .	3620
6.787	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	
	Class Reference . . . . .	3620
6.787.1	Detailed Description . . . . .	3621
6.787.2	Constructor & Destructor Documentation . . . . .	3621
6.787.2.1	SubscriptionInfoMarshaller . . . . .	3621
6.787.2.2	~SubscriptionInfoMarshaller . . . . .	3621
6.787.3	Member Function Documentation . . . . .	3621
6.787.3.1	createObject . . . . .	3621
6.787.3.2	getDataStructureType . . . . .	3622
6.787.3.3	looseMarshal . . . . .	3622
6.787.3.4	looseUnmarshal . . . . .	3622
6.787.3.5	tightMarshal1 . . . . .	3623
6.787.3.6	tightMarshal2 . . . . .	3623
6.787.3.7	tightUnmarshal . . . . .	3624
6.788	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	
	Class Reference . . . . .	3624
6.788.1	Detailed Description . . . . .	3625
6.788.2	Constructor & Destructor Documentation . . . . .	3625

6.788.2.1 SubscriptionInfoMarshaller . . . . .	3625
6.788.2.2 ~SubscriptionInfoMarshaller . . . . .	3625
6.788.3 Member Function Documentation . . . . .	3625
6.788.3.1 createObject . . . . .	3625
6.788.3.2 getDataStructureType . . . . .	3626
6.788.3.3 looseMarshal . . . . .	3626
6.788.3.4 looseUnmarshal . . . . .	3626
6.788.3.5 tightMarshal1 . . . . .	3627
6.788.3.6 tightMarshal2 . . . . .	3627
6.788.3.7 tightUnmarshal . . . . .	3628
6.789activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	
Class Reference . . . . .	3628
6.789.1 Detailed Description . . . . .	3629
6.789.2 Constructor & Destructor Documentation . . . . .	3629
6.789.2.1 SubscriptionInfoMarshaller . . . . .	3629
6.789.2.2 ~SubscriptionInfoMarshaller . . . . .	3629
6.789.3 Member Function Documentation . . . . .	3629
6.789.3.1 createObject . . . . .	3629
6.789.3.2 getDataStructureType . . . . .	3630
6.789.3.3 looseMarshal . . . . .	3630
6.789.3.4 looseUnmarshal . . . . .	3630
6.789.3.5 tightMarshal1 . . . . .	3631
6.789.3.6 tightMarshal2 . . . . .	3631
6.789.3.7 tightUnmarshal . . . . .	3632
6.790activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	
Class Reference . . . . .	3632
6.790.1 Detailed Description . . . . .	3633
6.790.2 Constructor & Destructor Documentation . . . . .	3633
6.790.2.1 SubscriptionInfoMarshaller . . . . .	3633
6.790.2.2 ~SubscriptionInfoMarshaller . . . . .	3633
6.790.3 Member Function Documentation . . . . .	3633
6.790.3.1 createObject . . . . .	3633
6.790.3.2 getDataStructureType . . . . .	3634
6.790.3.3 looseMarshal . . . . .	3634

6.790.3.4 looseUnmarshal . . . . .	3634
6.790.3.5 tightMarshal1 . . . . .	3635
6.790.3.6 tightMarshal2 . . . . .	3635
6.790.3.7 tightUnmarshal . . . . .	3636
6.791 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller	
Class Reference . . . . .	3636
6.791.1 Detailed Description . . . . .	3637
6.791.2 Constructor & Destructor Documentation . . . . .	3637
6.791.2.1 SubscriptionInfoMarshaller . . . . .	3637
6.791.2.2 ~SubscriptionInfoMarshaller . . . . .	3637
6.791.3 Member Function Documentation . . . . .	3637
6.791.3.1 createObject . . . . .	3637
6.791.3.2 getDataStructureType . . . . .	3638
6.791.3.3 looseMarshal . . . . .	3638
6.791.3.4 looseUnmarshal . . . . .	3638
6.791.3.5 tightMarshal1 . . . . .	3639
6.791.3.6 tightMarshal2 . . . . .	3639
6.791.3.7 tightUnmarshal . . . . .	3640
6.792 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	
Class Reference . . . . .	3640
6.792.1 Detailed Description . . . . .	3641
6.792.2 Constructor & Destructor Documentation . . . . .	3641
6.792.2.1 SubscriptionInfoMarshaller . . . . .	3641
6.792.2.2 ~SubscriptionInfoMarshaller . . . . .	3641
6.792.3 Member Function Documentation . . . . .	3641
6.792.3.1 createObject . . . . .	3641
6.792.3.2 getDataStructureType . . . . .	3642
6.792.3.3 looseMarshal . . . . .	3642
6.792.3.4 looseUnmarshal . . . . .	3642
6.792.3.5 tightMarshal1 . . . . .	3643
6.792.3.6 tightMarshal2 . . . . .	3643
6.792.3.7 tightUnmarshal . . . . .	3644
6.793 decaf::util::concurrent::Synchronizable Class Reference . . . . .	3644
6.793.1 Detailed Description . . . . .	3645

6.793.2 Constructor & Destructor Documentation . . . . .	3645
6.793.2.1 ~Synchronizable . . . . .	3645
6.793.3 Member Function Documentation . . . . .	3645
6.793.3.1 lock . . . . .	3645
6.793.3.2 notify . . . . .	3646
6.793.3.3 notifyAll . . . . .	3647
6.793.3.4 tryLock . . . . .	3649
6.793.3.5 unlock . . . . .	3650
6.793.3.6 wait . . . . .	3651
6.793.3.7 wait . . . . .	3652
6.793.3.8 wait . . . . .	3653
6.794decaf::internal::util::concurrent::SynchronizableImpl Class Reference . . . . .	3655
6.794.1 Detailed Description . . . . .	3655
6.794.2 Constructor & Destructor Documentation . . . . .	3656
6.794.2.1 SynchronizableImpl . . . . .	3656
6.794.2.2 ~SynchronizableImpl . . . . .	3656
6.794.3 Member Function Documentation . . . . .	3656
6.794.3.1 lock . . . . .	3656
6.794.3.2 notify . . . . .	3656
6.794.3.3 notifyAll . . . . .	3656
6.794.3.4 tryLock . . . . .	3657
6.794.3.5 unlock . . . . .	3657
6.794.3.6 wait . . . . .	3657
6.794.3.7 wait . . . . .	3658
6.794.3.8 wait . . . . .	3658
6.795activemq::core::Synchronization Class Reference . . . . .	3659
6.795.1 Detailed Description . . . . .	3659
6.795.2 Constructor & Destructor Documentation . . . . .	3659
6.795.2.1 ~Synchronization . . . . .	3659
6.795.3 Member Function Documentation . . . . .	3659
6.795.3.1 afterCommit . . . . .	3659
6.795.3.2 afterRollback . . . . .	3659
6.795.3.3 beforeEnd . . . . .	3660

6.796decaf::util::concurrent::SynchronousQueue< E > Class Template Reference . . . . .	3660
6.796.1 Detailed Description . . . . .	3662
6.796.2 Constructor & Destructor Documentation . . . . .	3662
6.796.2.1 SynchronousQueue . . . . .	3662
6.796.2.2 ~SynchronousQueue . . . . .	3662
6.796.3 Member Function Documentation . . . . .	3662
6.796.3.1 clear . . . . .	3662
6.796.3.2 contains . . . . .	3663
6.796.3.3 containsAll . . . . .	3663
6.796.3.4 drainTo . . . . .	3663
6.796.3.5 drainTo . . . . .	3664
6.796.3.6 equals . . . . .	3665
6.796.3.7 isEmpty . . . . .	3665
6.796.3.8 iterator . . . . .	3665
6.796.3.9 iterator . . . . .	3665
6.796.3.10offer . . . . .	3666
6.796.3.11offer . . . . .	3666
6.796.3.12peek . . . . .	3667
6.796.3.13poll . . . . .	3667
6.796.3.14poll . . . . .	3667
6.796.3.15put . . . . .	3668
6.796.3.16remainingCapacity . . . . .	3668
6.796.3.17remove . . . . .	3668
6.796.3.18removeAll . . . . .	3669
6.796.3.19retainAll . . . . .	3669
6.796.3.20size . . . . .	3669
6.796.3.21take . . . . .	3669
6.796.3.22toArray . . . . .	3670
6.797decaf::lang::System Class Reference . . . . .	3670
6.797.1 Detailed Description . . . . .	3672
6.797.2 Constructor & Destructor Documentation . . . . .	3672
6.797.2.1 System . . . . .	3672
6.797.2.2 ~System . . . . .	3672

6.797.3 Member Function Documentation . . . . .	3672
6.797.3.1 arraycopy . . . . .	3672
6.797.3.2 arraycopy . . . . .	3672
6.797.3.3 arraycopy . . . . .	3673
6.797.3.4 arraycopy . . . . .	3673
6.797.3.5 availableProcessors . . . . .	3674
6.797.3.6 clearProperty . . . . .	3674
6.797.3.7 currentTimeMillis . . . . .	3674
6.797.3.8 getenv . . . . .	3675
6.797.3.9 getenv . . . . .	3675
6.797.3.10getProperties . . . . .	3675
6.797.3.11getProperty . . . . .	3675
6.797.3.12getProperty . . . . .	3676
6.797.3.13nanoTime . . . . .	3676
6.797.3.14setenv . . . . .	3677
6.797.3.15setProperty . . . . .	3677
6.797.3.16unsetenv . . . . .	3678
6.797.4 Friends And Related Function Documentation . . . . .	3678
6.797.4.1 decaf::lang::Runtime . . . . .	3678
6.798activemq::threads::Task Class Reference . . . . .	3678
6.798.1 Detailed Description . . . . .	3678
6.798.2 Constructor & Destructor Documentation . . . . .	3679
6.798.2.1 ~Task . . . . .	3679
6.798.3 Member Function Documentation . . . . .	3679
6.798.3.1 iterate . . . . .	3679
6.799decaf::util::concurrent::TaskListener Class Reference . . . . .	3679
6.799.1 Constructor & Destructor Documentation . . . . .	3679
6.799.1.1 ~TaskListener . . . . .	3679
6.799.2 Member Function Documentation . . . . .	3679
6.799.2.1 onTaskComplete . . . . .	3680
6.799.2.2 onTaskException . . . . .	3680
6.800activemq::threads::TaskRunner Class Reference . . . . .	3680
6.800.1 Constructor & Destructor Documentation . . . . .	3681
6.800.1.1 ~TaskRunner . . . . .	3681

6.800.2 Member Function Documentation . . . . .	3681
6.800.2.1 shutdown . . . . .	3681
6.800.2.2 shutdown . . . . .	3681
6.800.2.3 wakeup . . . . .	3681
6.801 decaf::internal::net::tcp::TcpSocket Class Reference . . . . .	3681
6.801.1 Detailed Description . . . . .	3684
6.801.2 Constructor & Destructor Documentation . . . . .	3685
6.801.2.1 TcpSocket . . . . .	3685
6.801.2.2 ~TcpSocket . . . . .	3685
6.801.3 Member Function Documentation . . . . .	3685
6.801.3.1 accept . . . . .	3685
6.801.3.2 available . . . . .	3685
6.801.3.3 bind . . . . .	3685
6.801.3.4 checkResult . . . . .	3686
6.801.3.5 close . . . . .	3686
6.801.3.6 connect . . . . .	3686
6.801.3.7 create . . . . .	3686
6.801.3.8 getInputStream . . . . .	3687
6.801.3.9 getLocalAddress . . . . .	3687
6.801.3.10getOption . . . . .	3687
6.801.3.11getOutputStream . . . . .	3688
6.801.3.12getSocketHandle . . . . .	3688
6.801.3.13sClosed . . . . .	3688
6.801.3.14sConnected . . . . .	3688
6.801.3.15listen . . . . .	3688
6.801.3.16read . . . . .	3689
6.801.3.17setOption . . . . .	3689
6.801.3.18shutdownInput . . . . .	3690
6.801.3.19shutdownOutput . . . . .	3690
6.801.3.20write . . . . .	3690
6.802 decaf::internal::net::tcp::TcpSocketInputStream Class Reference . . . . .	3691
6.802.1 Detailed Description . . . . .	3692
6.802.2 Constructor & Destructor Documentation . . . . .	3692
6.802.2.1 TcpSocketInputStream . . . . .	3692

6.802.2.2 ~TcpSocketInputStream . . . . .	3692
6.802.3 Member Function Documentation . . . . .	3692
6.802.3.1 available . . . . .	3692
6.802.3.2 close . . . . .	3693
6.802.3.3 doReadArrayBounded . . . . .	3693
6.802.3.4 doReadByte . . . . .	3693
6.802.3.5 skip . . . . .	3693
6.803decaf::internal::net::tcp::TcpSocketOutputStream Class Reference . . . . .	3694
6.803.1 Detailed Description . . . . .	3695
6.803.2 Constructor & Destructor Documentation . . . . .	3695
6.803.2.1 TcpSocketOutputStream . . . . .	3695
6.803.2.2 ~TcpSocketOutputStream . . . . .	3695
6.803.3 Member Function Documentation . . . . .	3695
6.803.3.1 close . . . . .	3695
6.803.3.2 doWriteArrayBounded . . . . .	3695
6.803.3.3 doWriteByte . . . . .	3696
6.804activemq::transport::tcp::TcpTransport Class Reference . . . . .	3696
6.804.1 Detailed Description . . . . .	3697
6.804.2 Constructor & Destructor Documentation . . . . .	3697
6.804.2.1 TcpTransport . . . . .	3697
6.804.2.2 ~TcpTransport . . . . .	3697
6.804.3 Member Function Documentation . . . . .	3697
6.804.3.1 close . . . . .	3697
6.804.3.2 configureSocket . . . . .	3697
6.804.3.3 connect . . . . .	3698
6.804.3.4 createSocket . . . . .	3698
6.804.3.5 isClosed . . . . .	3698
6.804.3.6 isConnected . . . . .	3699
6.804.3.7 isFaultTolerant . . . . .	3699
6.805activemq::transport::tcp::TcpTransportFactory Class Reference . . . . .	3699
6.805.1 Detailed Description . . . . .	3700
6.805.2 Constructor & Destructor Documentation . . . . .	3700
6.805.2.1 ~TcpTransportFactory . . . . .	3700
6.805.3 Member Function Documentation . . . . .	3700



6.805.3.1 create . . . . .	3700
6.805.3.2 createComposite . . . . .	3701
6.805.3.3 doCreateComposite . . . . .	3701
6.806cms::TemporaryQueue Class Reference . . . . .	3701
6.806.1 Detailed Description . . . . .	3702
6.806.2 Constructor & Destructor Documentation . . . . .	3702
6.806.2.1 ~TemporaryQueue . . . . .	3702
6.806.3 Member Function Documentation . . . . .	3702
6.806.3.1 destroy . . . . .	3702
6.806.3.2 getQueueName . . . . .	3703
6.807cms::TemporaryTopic Class Reference . . . . .	3703
6.807.1 Detailed Description . . . . .	3703
6.807.2 Constructor & Destructor Documentation . . . . .	3704
6.807.2.1 ~TemporaryTopic . . . . .	3704
6.807.3 Member Function Documentation . . . . .	3704
6.807.3.1 destroy . . . . .	3704
6.807.3.2 getTopicName . . . . .	3704
6.808cms::TextMessage Class Reference . . . . .	3704
6.808.1 Detailed Description . . . . .	3705
6.808.2 Constructor & Destructor Documentation . . . . .	3705
6.808.2.1 ~TextMessage . . . . .	3705
6.808.3 Member Function Documentation . . . . .	3705
6.808.3.1 getText . . . . .	3705
6.808.3.2 setText . . . . .	3706
6.808.3.3 setText . . . . .	3706
6.809decaf::lang::Thread Class Reference . . . . .	3707
6.809.1 Detailed Description . . . . .	3709
6.809.2 Member Enumeration Documentation . . . . .	3709
6.809.2.1 State . . . . .	3710
6.809.3 Constructor & Destructor Documentation . . . . .	3710
6.809.3.1 Thread . . . . .	3710
6.809.3.2 Thread . . . . .	3710
6.809.3.3 Thread . . . . .	3710
6.809.3.4 Thread . . . . .	3711

6.809.3.5 ~Thread . . . . .	3711
6.809.4 Member Function Documentation . . . . .	3711
6.809.4.1 currentThread . . . . .	3711
6.809.4.2 getId . . . . .	3711
6.809.4.3 getName . . . . .	3711
6.809.4.4 getPriority . . . . .	3712
6.809.4.5 getState . . . . .	3712
6.809.4.6 getUncaughtExceptionHandler . . . . .	3712
6.809.4.7 isAlive . . . . .	3712
6.809.4.8 join . . . . .	3712
6.809.4.9 join . . . . .	3713
6.809.4.10join . . . . .	3713
6.809.4.11run . . . . .	3713
6.809.4.12setName . . . . .	3713
6.809.4.13setPriority . . . . .	3714
6.809.4.14setUncaughtExceptionHandler . . . . .	3714
6.809.4.15sleep . . . . .	3714
6.809.4.16sleep . . . . .	3715
6.809.4.17start . . . . .	3715
6.809.4.18toString . . . . .	3715
6.809.4.19yield . . . . .	3715
6.809.5 Friends And Related Function Documentation . . . . .	3716
6.809.5.1 decaf::lang::Runtime . . . . .	3716
6.809.5.2 decaf::util::concurrent::locks::LockSupport . . . . .	3716
6.809.6 Field Documentation . . . . .	3716
6.809.6.1 MAX_PRIORITY . . . . .	3716
6.809.6.2 MIN_PRIORITY . . . . .	3716
6.809.6.3 NORM_PRIORITY . . . . .	3716
6.810decaf::util::concurrent::ThreadFactory Class Reference . . . . .	3716
6.810.1 Detailed Description . . . . .	3716
6.810.2 Constructor & Destructor Documentation . . . . .	3717
6.810.2.1 ~ThreadFactory . . . . .	3717
6.810.3 Member Function Documentation . . . . .	3717
6.810.3.1 newThread . . . . .	3717

6.811	decaf::lang::ThreadGroup Class Reference . . . . .	3717
6.811.1	Detailed Description . . . . .	3718
6.811.2	Constructor & Destructor Documentation . . . . .	3718
6.811.2.1	ThreadGroup . . . . .	3718
6.811.2.2	~ThreadGroup . . . . .	3718
6.812	decaf::util::concurrent::ThreadPool Class Reference . . . . .	3718
6.812.1	Detailed Description . . . . .	3719
6.812.2	Member Typedef Documentation . . . . .	3720
6.812.2.1	Task . . . . .	3720
6.812.3	Constructor & Destructor Documentation . . . . .	3720
6.812.3.1	ThreadPool . . . . .	3720
6.812.3.2	~ThreadPool . . . . .	3720
6.812.4	Member Function Documentation . . . . .	3720
6.812.4.1	deQueueTask . . . . .	3720
6.812.4.2	getBacklog . . . . .	3720
6.812.4.3	getBlockSize . . . . .	3721
6.812.4.4	getFreeThreadCount . . . . .	3721
6.812.4.5	getInstance . . . . .	3721
6.812.4.6	getMaxThreads . . . . .	3721
6.812.4.7	getPoolSize . . . . .	3721
6.812.4.8	onTaskCompleted . . . . .	3722
6.812.4.9	onTaskException . . . . .	3722
6.812.4.10	onTaskStarted . . . . .	3722
6.812.4.11	queueTask . . . . .	3722
6.812.4.12	reserve . . . . .	3723
6.812.4.13	setBlockSize . . . . .	3723
6.812.4.14	setMaxThreads . . . . .	3723
6.812.5	Field Documentation . . . . .	3723
6.812.5.1	DEFAULT_MAX_BLOCK_SIZE . . . . .	3724
6.812.5.2	DEFAULT_MAX_POOL_SIZE . . . . .	3724
6.813	decaf::lang::Throwable Class Reference . . . . .	3724
6.813.1	Detailed Description . . . . .	3725
6.813.2	Constructor & Destructor Documentation . . . . .	3725
6.813.2.1	Throwable . . . . .	3725

6.813.2.2 ~Throwable . . . . .	3725
6.813.3 Member Function Documentation . . . . .	3725
6.813.3.1 clone . . . . .	3725
6.813.3.2 getCause . . . . .	3726
6.813.3.3 getMessage . . . . .	3726
6.813.3.4 getStackTrace . . . . .	3726
6.813.3.5 getStackTraceString . . . . .	3727
6.813.3.6 initCause . . . . .	3727
6.813.3.7 printStackTrace . . . . .	3727
6.813.3.8 printStackTrace . . . . .	3727
6.813.3.9 setMark . . . . .	3728
6.814decaf::util::concurrent::TimeoutException Class Reference . . . . .	3728
6.814.1 Constructor & Destructor Documentation . . . . .	3729
6.814.1.1 TimeoutException . . . . .	3729
6.814.1.2 TimeoutException . . . . .	3729
6.814.1.3 TimeoutException . . . . .	3729
6.814.1.4 TimeoutException . . . . .	3729
6.814.1.5 TimeoutException . . . . .	3729
6.814.1.6 TimeoutException . . . . .	3730
6.814.1.7 ~TimeoutException . . . . .	3730
6.814.2 Member Function Documentation . . . . .	3730
6.814.2.1 clone . . . . .	3730
6.815decaf::util::Timer Class Reference . . . . .	3730
6.815.1 Detailed Description . . . . .	3732
6.815.2 Constructor & Destructor Documentation . . . . .	3732
6.815.2.1 Timer . . . . .	3733
6.815.2.2 ~Timer . . . . .	3733
6.815.3 Member Function Documentation . . . . .	3733
6.815.3.1 cancel . . . . .	3733
6.815.3.2 purge . . . . .	3733
6.815.3.3 schedule . . . . .	3733
6.815.3.4 schedule . . . . .	3734
6.815.3.5 schedule . . . . .	3735
6.815.3.6 schedule . . . . .	3736

6.815.3.7 schedule . . . . .	3736
6.815.3.8 schedule . . . . .	3737
6.815.3.9 schedule . . . . .	3738
6.815.3.10 schedule . . . . .	3739
6.815.3.11 scheduleAtFixedRate . . . . .	3739
6.815.3.12 scheduleAtFixedRate . . . . .	3740
6.815.3.13 scheduleAtFixedRate . . . . .	3741
6.815.3.14 scheduleAtFixedRate . . . . .	3742
6.816 decaf::util::TimerTask Class Reference . . . . .	3743
6.816.1 Detailed Description . . . . .	3743
6.816.2 Constructor & Destructor Documentation . . . . .	3743
6.816.2.1 TimerTask . . . . .	3743
6.816.2.2 ~TimerTask . . . . .	3744
6.816.3 Member Function Documentation . . . . .	3744
6.816.3.1 cancel . . . . .	3744
6.816.3.2 getWhen . . . . .	3744
6.816.3.3 isScheduled . . . . .	3744
6.816.3.4 scheduledExecutionTime . . . . .	3744
6.816.3.5 setScheduledTime . . . . .	3745
6.816.4 Friends And Related Function Documentation . . . . .	3745
6.816.4.1 decaf::internal::util::TimerTaskHeap . . . . .	3745
6.816.4.2 Timer . . . . .	3745
6.816.4.3 TimerImpl . . . . .	3745
6.817 decaf::internal::util::TimerTaskHeap Class Reference . . . . .	3745
6.817.1 Detailed Description . . . . .	3746
6.817.2 Constructor & Destructor Documentation . . . . .	3746
6.817.2.1 TimerTaskHeap . . . . .	3746
6.817.2.2 ~TimerTaskHeap . . . . .	3746
6.817.3 Member Function Documentation . . . . .	3746
6.817.3.1 adjustMinimum . . . . .	3746
6.817.3.2 deleteIfCancelled . . . . .	3746
6.817.3.3 find . . . . .	3746
6.817.3.4 insert . . . . .	3746
6.817.3.5 isEmpty . . . . .	3747

6.817.3.6 peek . . . . .	3747
6.817.3.7 remove . . . . .	3747
6.817.3.8 reset . . . . .	3747
6.817.3.9 size . . . . .	3747
6.818decaf::util::concurrent::TimeUnit Class Reference . . . . .	3748
6.818.1 Detailed Description . . . . .	3749
6.818.2 Constructor & Destructor Documentation . . . . .	3750
6.818.2.1 TimeUnit . . . . .	3750
6.818.2.2 ~TimeUnit . . . . .	3750
6.818.3 Member Function Documentation . . . . .	3750
6.818.3.1 compareTo . . . . .	3750
6.818.3.2 convert . . . . .	3751
6.818.3.3 equals . . . . .	3751
6.818.3.4 operator< . . . . .	3751
6.818.3.5 operator== . . . . .	3752
6.818.3.6 sleep . . . . .	3752
6.818.3.7 timedJoin . . . . .	3752
6.818.3.8 timedWait . . . . .	3753
6.818.3.9 toDays . . . . .	3754
6.818.3.10toHours . . . . .	3754
6.818.3.11toMicros . . . . .	3754
6.818.3.12toMillis . . . . .	3755
6.818.3.13toMinutes . . . . .	3755
6.818.3.14toNanos . . . . .	3755
6.818.3.15toSeconds . . . . .	3756
6.818.3.16toString . . . . .	3756
6.818.3.17valueOf . . . . .	3756
6.818.4 Field Documentation . . . . .	3757
6.818.4.1 DAYS . . . . .	3757
6.818.4.2 HOURS . . . . .	3757
6.818.4.3 MICROSECONDS . . . . .	3757
6.818.4.4 MILLISECONDS . . . . .	3757
6.818.4.5 MINUTES . . . . .	3757
6.818.4.6 NANOSECONDS . . . . .	3757

6.818.4.7 SECONDS . . . . .	3757
6.818.4.8 values . . . . .	3757
6.819cms::Topic Class Reference . . . . .	3757
6.819.1 Detailed Description . . . . .	3758
6.819.2 Constructor & Destructor Documentation . . . . .	3758
6.819.2.1 ~Topic . . . . .	3758
6.819.3 Member Function Documentation . . . . .	3758
6.819.3.1 getTopicName . . . . .	3758
6.820activemq::state::Tracked Class Reference . . . . .	3758
6.820.1 Constructor & Destructor Documentation . . . . .	3759
6.820.1.1 Tracked . . . . .	3759
6.820.1.2 Tracked . . . . .	3759
6.820.1.3 ~Tracked . . . . .	3759
6.820.2 Member Function Documentation . . . . .	3759
6.820.2.1 isWaitingForResponse . . . . .	3759
6.820.2.2 onResponse . . . . .	3759
6.821activemq::commands::TransactionId Class Reference . . . . .	3759
6.821.1 Member Typedef Documentation . . . . .	3760
6.821.1.1 COMPARATOR . . . . .	3760
6.821.2 Constructor & Destructor Documentation . . . . .	3760
6.821.2.1 TransactionId . . . . .	3760
6.821.2.2 TransactionId . . . . .	3760
6.821.2.3 ~TransactionId . . . . .	3760
6.821.3 Member Function Documentation . . . . .	3761
6.821.3.1 cloneDataStructure . . . . .	3761
6.821.3.2 compareTo . . . . .	3761
6.821.3.3 copyDataStructure . . . . .	3761
6.821.3.4 equals . . . . .	3761
6.821.3.5 equals . . . . .	3762
6.821.3.6 getDataStructureType . . . . .	3762
6.821.3.7 operator< . . . . .	3762
6.821.3.8 operator= . . . . .	3762
6.821.3.9 operator== . . . . .	3762
6.821.3.10toString . . . . .	3762

6.821.4 Field Documentation . . . . .	3762
6.821.4.1 ID_TRANSACTIONID . . . . .	3762
6.822activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Reference . . . . .	3763
6.822.1 Detailed Description . . . . .	3763
6.822.2 Constructor & Destructor Documentation . . . . .	3763
6.822.2.1 TransactionIdMarshaller . . . . .	3764
6.822.2.2 ~TransactionIdMarshaller . . . . .	3764
6.822.3 Member Function Documentation . . . . .	3764
6.822.3.1 looseMarshal . . . . .	3764
6.822.3.2 looseUnmarshal . . . . .	3764
6.822.3.3 tightMarshal1 . . . . .	3765
6.822.3.4 tightMarshal2 . . . . .	3765
6.822.3.5 tightUnmarshal . . . . .	3766
6.823activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference . . . . .	3766
6.823.1 Detailed Description . . . . .	3767
6.823.2 Constructor & Destructor Documentation . . . . .	3767
6.823.2.1 TransactionIdMarshaller . . . . .	3767
6.823.2.2 ~TransactionIdMarshaller . . . . .	3767
6.823.3 Member Function Documentation . . . . .	3767
6.823.3.1 looseMarshal . . . . .	3767
6.823.3.2 looseUnmarshal . . . . .	3768
6.823.3.3 tightMarshal1 . . . . .	3768
6.823.3.4 tightMarshal2 . . . . .	3769
6.823.3.5 tightUnmarshal . . . . .	3769
6.824activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference . . . . .	3770
6.824.1 Detailed Description . . . . .	3771
6.824.2 Constructor & Destructor Documentation . . . . .	3771
6.824.2.1 TransactionIdMarshaller . . . . .	3771
6.824.2.2 ~TransactionIdMarshaller . . . . .	3771
6.824.3 Member Function Documentation . . . . .	3771
6.824.3.1 looseMarshal . . . . .	3771
6.824.3.2 looseUnmarshal . . . . .	3772



6.824.3.3 tightMarshal1 . . . . .	3772
6.824.3.4 tightMarshal2 . . . . .	3773
6.824.3.5 tightUnmarshal . . . . .	3773
6.825activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class	
Reference . . . . .	3774
6.825.1 Detailed Description . . . . .	3775
6.825.2 Constructor & Destructor Documentation . . . . .	3775
6.825.2.1 TransactionIdMarshaller . . . . .	3775
6.825.2.2 ~TransactionIdMarshaller . . . . .	3775
6.825.3 Member Function Documentation . . . . .	3775
6.825.3.1 looseMarshal . . . . .	3775
6.825.3.2 looseUnmarshal . . . . .	3775
6.825.3.3 tightMarshal1 . . . . .	3776
6.825.3.4 tightMarshal2 . . . . .	3776
6.825.3.5 tightUnmarshal . . . . .	3777
6.826activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class	
Reference . . . . .	3778
6.826.1 Detailed Description . . . . .	3778
6.826.2 Constructor & Destructor Documentation . . . . .	3778
6.826.2.1 TransactionIdMarshaller . . . . .	3779
6.826.2.2 ~TransactionIdMarshaller . . . . .	3779
6.826.3 Member Function Documentation . . . . .	3779
6.826.3.1 looseMarshal . . . . .	3779
6.826.3.2 looseUnmarshal . . . . .	3779
6.826.3.3 tightMarshal1 . . . . .	3780
6.826.3.4 tightMarshal2 . . . . .	3780
6.826.3.5 tightUnmarshal . . . . .	3781
6.827activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller Class	
Reference . . . . .	3781
6.827.1 Detailed Description . . . . .	3782
6.827.2 Constructor & Destructor Documentation . . . . .	3782
6.827.2.1 TransactionIdMarshaller . . . . .	3782
6.827.2.2 ~TransactionIdMarshaller . . . . .	3782
6.827.3 Member Function Documentation . . . . .	3782
6.827.3.1 looseMarshal . . . . .	3782

6.827.3.2 looseUnmarshal . . . . .	3783
6.827.3.3 tightMarshal1 . . . . .	3783
6.827.3.4 tightMarshal2 . . . . .	3784
6.827.3.5 tightUnmarshal . . . . .	3784
6.828activemq::commands::TransactionInfo Class Reference . . . . .	3785
6.828.1 Constructor & Destructor Documentation . . . . .	3786
6.828.1.1 TransactionInfo . . . . .	3786
6.828.1.2 ~TransactionInfo . . . . .	3786
6.828.2 Member Function Documentation . . . . .	3786
6.828.2.1 cloneDataStructure . . . . .	3786
6.828.2.2 copyDataStructure . . . . .	3787
6.828.2.3 equals . . . . .	3787
6.828.2.4 getConnectionId . . . . .	3787
6.828.2.5 getConnectionId . . . . .	3787
6.828.2.6 getDataStructureType . . . . .	3787
6.828.2.7 getTransactionId . . . . .	3787
6.828.2.8 getTransactionId . . . . .	3788
6.828.2.9 getType . . . . .	3788
6.828.2.10sTransactionInfo . . . . .	3788
6.828.2.11setConnectionId . . . . .	3788
6.828.2.12setTransactionId . . . . .	3788
6.828.2.13setType . . . . .	3788
6.828.2.14toString . . . . .	3788
6.828.2.15visit . . . . .	3788
6.828.3 Field Documentation . . . . .	3789
6.828.3.1 connectionId . . . . .	3789
6.828.3.2 ID_TRANSACTIONINFO . . . . .	3789
6.828.3.3 transactionId . . . . .	3789
6.828.3.4 type . . . . .	3789
6.829activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference . . . . .	3789
6.829.1 Detailed Description . . . . .	3790
6.829.2 Constructor & Destructor Documentation . . . . .	3790
6.829.2.1 TransactionInfoMarshaller . . . . .	3790

6.829.2.2 ~TransactionInfoMarshaller . . . . .	3790
6.829.3 Member Function Documentation . . . . .	3790
6.829.3.1 createObject . . . . .	3790
6.829.3.2 getDataStructureType . . . . .	3790
6.829.3.3 looseMarshal . . . . .	3791
6.829.3.4 looseUnmarshal . . . . .	3791
6.829.3.5 tightMarshal1 . . . . .	3792
6.829.3.6 tightMarshal2 . . . . .	3792
6.829.3.7 tightUnmarshal . . . . .	3793
6.830activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference . . . . .	3793
6.830.1 Detailed Description . . . . .	3794
6.830.2 Constructor & Destructor Documentation . . . . .	3794
6.830.2.1 TransactionInfoMarshaller . . . . .	3794
6.830.2.2 ~TransactionInfoMarshaller . . . . .	3794
6.830.3 Member Function Documentation . . . . .	3794
6.830.3.1 createObject . . . . .	3794
6.830.3.2 getDataStructureType . . . . .	3795
6.830.3.3 looseMarshal . . . . .	3795
6.830.3.4 looseUnmarshal . . . . .	3795
6.830.3.5 tightMarshal1 . . . . .	3796
6.830.3.6 tightMarshal2 . . . . .	3796
6.830.3.7 tightUnmarshal . . . . .	3797
6.831activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference . . . . .	3797
6.831.1 Detailed Description . . . . .	3798
6.831.2 Constructor & Destructor Documentation . . . . .	3798
6.831.2.1 TransactionInfoMarshaller . . . . .	3798
6.831.2.2 ~TransactionInfoMarshaller . . . . .	3798
6.831.3 Member Function Documentation . . . . .	3798
6.831.3.1 createObject . . . . .	3798
6.831.3.2 getDataStructureType . . . . .	3799
6.831.3.3 looseMarshal . . . . .	3799
6.831.3.4 looseUnmarshal . . . . .	3799

6.831.3.5 tightMarshal1 . . . . .	3800
6.831.3.6 tightMarshal2 . . . . .	3800
6.831.3.7 tightUnmarshal . . . . .	3801
6.832activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller Class Reference . . . . .	3801
6.832.1 Detailed Description . . . . .	3802
6.832.2 Constructor & Destructor Documentation . . . . .	3802
6.832.2.1 TransactionInfoMarshaller . . . . .	3802
6.832.2.2 ~TransactionInfoMarshaller . . . . .	3802
6.832.3 Member Function Documentation . . . . .	3802
6.832.3.1 createObject . . . . .	3802
6.832.3.2 getDataStructureType . . . . .	3803
6.832.3.3 looseMarshal . . . . .	3803
6.832.3.4 looseUnmarshal . . . . .	3803
6.832.3.5 tightMarshal1 . . . . .	3804
6.832.3.6 tightMarshal2 . . . . .	3804
6.832.3.7 tightUnmarshal . . . . .	3805
6.833activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference . . . . .	3805
6.833.1 Detailed Description . . . . .	3806
6.833.2 Constructor & Destructor Documentation . . . . .	3806
6.833.2.1 TransactionInfoMarshaller . . . . .	3806
6.833.2.2 ~TransactionInfoMarshaller . . . . .	3806
6.833.3 Member Function Documentation . . . . .	3806
6.833.3.1 createObject . . . . .	3806
6.833.3.2 getDataStructureType . . . . .	3807
6.833.3.3 looseMarshal . . . . .	3807
6.833.3.4 looseUnmarshal . . . . .	3807
6.833.3.5 tightMarshal1 . . . . .	3808
6.833.3.6 tightMarshal2 . . . . .	3808
6.833.3.7 tightUnmarshal . . . . .	3809
6.834activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference . . . . .	3809
6.834.1 Detailed Description . . . . .	3810
6.834.2 Constructor & Destructor Documentation . . . . .	3810

6.834.2.1 TransactionInfoMarshaller . . . . .	3810
6.834.2.2 ~TransactionInfoMarshaller . . . . .	3810
6.834.3 Member Function Documentation . . . . .	3810
6.834.3.1 createObject . . . . .	3810
6.834.3.2 getDataStructureType . . . . .	3811
6.834.3.3 looseMarshal . . . . .	3811
6.834.3.4 looseUnmarshal . . . . .	3811
6.834.3.5 tightMarshal1 . . . . .	3812
6.834.3.6 tightMarshal2 . . . . .	3812
6.834.3.7 tightUnmarshal . . . . .	3813
6.835activemq::state::TransactionState Class Reference . . . . .	3813
6.835.1 Constructor & Destructor Documentation . . . . .	3814
6.835.1.1 TransactionState . . . . .	3814
6.835.1.2 ~TransactionState . . . . .	3814
6.835.2 Member Function Documentation . . . . .	3814
6.835.2.1 addCommand . . . . .	3814
6.835.2.2 addProducerState . . . . .	3814
6.835.2.3 checkShutdown . . . . .	3814
6.835.2.4 getCommands . . . . .	3814
6.835.2.5 getId . . . . .	3814
6.835.2.6 getPreparedResult . . . . .	3814
6.835.2.7 getProducerStates . . . . .	3814
6.835.2.8 isPrepared . . . . .	3814
6.835.2.9 setPrepared . . . . .	3814
6.835.2.10setPreparedResult . . . . .	3814
6.835.2.11shutdown . . . . .	3814
6.835.2.12toString . . . . .	3814
6.836decaf::internal::util::concurrent::Transferer< E > Class Template Refer- ence . . . . .	3815
6.836.1 Detailed Description . . . . .	3815
6.837decaf::internal::util::concurrent::TransferQueue< E > Class Template Ref- erence . . . . .	3815
6.837.1 Detailed Description . . . . .	3816
6.837.2 Constructor & Destructor Documentation . . . . .	3816

6.837.2.1 TransferQueue . . . . .	3816
6.837.2.2 ~TransferQueue . . . . .	3816
6.837.3 Member Function Documentation . . . . .	3816
6.837.3.1 transfer . . . . .	3816
6.837.3.2 transfer . . . . .	3817
6.838decaf::internal::util::concurrent::TransferStack< E > Class Template Reference . . . . .	3817
6.838.1 Constructor & Destructor Documentation . . . . .	3818
6.838.1.1 TransferStack . . . . .	3818
6.838.1.2 ~TransferStack . . . . .	3818
6.838.2 Member Function Documentation . . . . .	3818
6.838.2.1 transfer . . . . .	3818
6.838.2.2 transfer . . . . .	3818
6.839activemq::transport::Transport Class Reference . . . . .	3819
6.839.1 Detailed Description . . . . .	3820
6.839.2 Constructor & Destructor Documentation . . . . .	3820
6.839.2.1 ~Transport . . . . .	3820
6.839.3 Member Function Documentation . . . . .	3820
6.839.3.1 getRemoteAddress . . . . .	3821
6.839.3.2 getTransportListener . . . . .	3821
6.839.3.3 isClosed . . . . .	3821
6.839.3.4 isConnected . . . . .	3821
6.839.3.5 isFaultTolerant . . . . .	3822
6.839.3.6 narrow . . . . .	3822
6.839.3.7 oneway . . . . .	3822
6.839.3.8 reconnect . . . . .	3823
6.839.3.9 request . . . . .	3823
6.839.3.10request . . . . .	3824
6.839.3.11setTransportListener . . . . .	3824
6.839.3.12setWireFormat . . . . .	3824
6.839.3.13start . . . . .	3825
6.839.3.14stop . . . . .	3825
6.840activemq::transport::TransportFactory Class Reference . . . . .	3825
6.840.1 Detailed Description . . . . .	3826

6.840.2 Constructor & Destructor Documentation . . . . .	3826
6.840.2.1 ~TransportFactory . . . . .	3826
6.840.3 Member Function Documentation . . . . .	3826
6.840.3.1 create . . . . .	3826
6.840.3.2 createComposite . . . . .	3827
6.841 activemq::transport::TransportFilter Class Reference . . . . .	3827
6.841.1 Detailed Description . . . . .	3829
6.841.2 Constructor & Destructor Documentation . . . . .	3829
6.841.2.1 TransportFilter . . . . .	3829
6.841.2.2 ~TransportFilter . . . . .	3829
6.841.3 Member Function Documentation . . . . .	3829
6.841.3.1 close . . . . .	3829
6.841.3.2 fire . . . . .	3830
6.841.3.3 fire . . . . .	3830
6.841.3.4 getRemoteAddress . . . . .	3830
6.841.3.5 getTransportListener . . . . .	3830
6.841.3.6 isClosed . . . . .	3830
6.841.3.7 isConnected . . . . .	3831
6.841.3.8 isFaultTolerant . . . . .	3831
6.841.3.9 narrow . . . . .	3831
6.841.3.10 onCommand . . . . .	3832
6.841.3.11 oneway . . . . .	3832
6.841.3.12 onException . . . . .	3832
6.841.3.13 reconnect . . . . .	3833
6.841.3.14 request . . . . .	3833
6.841.3.15 request . . . . .	3833
6.841.3.16 setTransportListener . . . . .	3834
6.841.3.17 setWireFormat . . . . .	3834
6.841.3.18 start . . . . .	3834
6.841.3.19 stop . . . . .	3835
6.841.3.20 transportInterrupted . . . . .	3835
6.841.3.21 transportResumed . . . . .	3835
6.841.4 Field Documentation . . . . .	3835
6.841.4.1 listener . . . . .	3835

6.841.4.2 next . . . . .	3835
6.842activemq::transport::TransportListener Class Reference . . . . .	3836
6.842.1 Detailed Description . . . . .	3836
6.842.2 Constructor & Destructor Documentation . . . . .	3836
6.842.2.1 ~TransportListener . . . . .	3836
6.842.3 Member Function Documentation . . . . .	3836
6.842.3.1 onCommand . . . . .	3836
6.842.3.2 onException . . . . .	3837
6.842.3.3 transportInterrupted . . . . .	3837
6.842.3.4 transportResumed . . . . .	3837
6.843activemq::transport::TransportRegistry Class Reference . . . . .	3837
6.843.1 Detailed Description . . . . .	3838
6.843.2 Constructor & Destructor Documentation . . . . .	3838
6.843.2.1 ~TransportRegistry . . . . .	3838
6.843.3 Member Function Documentation . . . . .	3838
6.843.3.1 findFactory . . . . .	3838
6.843.3.2 getInstance . . . . .	3839
6.843.3.3 getTransportNames . . . . .	3839
6.843.3.4 registerFactory . . . . .	3839
6.843.3.5 unregisterFactory . . . . .	3840
6.844tree_desc_s Struct Reference . . . . .	3840
6.844.1 Field Documentation . . . . .	3840
6.844.1.1 dyn_tree . . . . .	3840
6.844.1.2 max_code . . . . .	3840
6.844.1.3 stat_desc . . . . .	3840
6.845decaf::lang::Thread::UncaughtExceptionHandler Class Reference . . . . .	3841
6.845.1 Detailed Description . . . . .	3841
6.845.2 Constructor & Destructor Documentation . . . . .	3841
6.845.2.1 ~UncaughtExceptionHandler . . . . .	3841
6.845.3 Member Function Documentation . . . . .	3841
6.845.3.1 uncaughtException . . . . .	3841
6.846decaf::net::UnknownHostException Class Reference . . . . .	3841
6.846.1 Constructor & Destructor Documentation . . . . .	3842
6.846.1.1 UnknownHostException . . . . .	3842



6.846.1.2 UnknownHostException . . . . .	3842
6.846.1.3 UnknownHostException . . . . .	3842
6.846.1.4 UnknownHostException . . . . .	3843
6.846.1.5 UnknownHostException . . . . .	3843
6.846.1.6 UnknownHostException . . . . .	3843
6.846.1.7 ~UnknownHostException . . . . .	3844
6.846.2 Member Function Documentation . . . . .	3844
6.846.2.1 clone . . . . .	3844
6.847decaf::net::UnknownServiceException Class Reference . . . . .	3844
6.847.1 Constructor & Destructor Documentation . . . . .	3845
6.847.1.1 UnknownServiceException . . . . .	3845
6.847.1.2 UnknownServiceException . . . . .	3845
6.847.1.3 UnknownServiceException . . . . .	3845
6.847.1.4 UnknownServiceException . . . . .	3845
6.847.1.5 UnknownServiceException . . . . .	3846
6.847.1.6 UnknownServiceException . . . . .	3846
6.847.1.7 ~UnknownServiceException . . . . .	3846
6.847.2 Member Function Documentation . . . . .	3846
6.847.2.1 clone . . . . .	3846
6.848decaf::io::UnsupportedEncodingException Class Reference . . . . .	3847
6.848.1 Detailed Description . . . . .	3847
6.848.2 Constructor & Destructor Documentation . . . . .	3847
6.848.2.1 UnsupportedEncodingException . . . . .	3847
6.848.2.2 UnsupportedEncodingException . . . . .	3848
6.848.2.3 UnsupportedEncodingException . . . . .	3848
6.848.2.4 UnsupportedEncodingException . . . . .	3848
6.848.2.5 UnsupportedEncodingException . . . . .	3848
6.848.2.6 UnsupportedEncodingException . . . . .	3849
6.848.2.7 ~UnsupportedEncodingException . . . . .	3849
6.848.3 Member Function Documentation . . . . .	3849
6.848.3.1 clone . . . . .	3849
6.849decaf::lang::exceptions::UnsupportedOperationException Class Reference	3849
6.849.1 Constructor & Destructor Documentation . . . . .	3850
6.849.1.1 UnsupportedOperationException . . . . .	3850

6.849.1.2	UnsupportedOperationException	3850
6.849.1.3	UnsupportedOperationException	3850
6.849.1.4	UnsupportedOperationException	3851
6.849.1.5	UnsupportedOperationException	3851
6.849.1.6	UnsupportedOperationException	3851
6.849.1.7	~UnsupportedOperationException	3851
6.849.2	Member Function Documentation	3851
6.849.2.1	clone	3852
6.850cms::	UnsupportedOperationException Class Reference	3852
6.850.1	Detailed Description	3852
6.850.2	Constructor & Destructor Documentation	3853
6.850.2.1	UnsupportedOperationException	3853
6.850.2.2	UnsupportedOperationException	3853
6.850.2.3	UnsupportedOperationException	3853
6.850.2.4	UnsupportedOperationException	3853
6.850.2.5	~UnsupportedOperationException	3853
6.851decaf::net::	URI Class Reference	3853
6.851.1	Detailed Description	3855
6.851.2	Constructor & Destructor Documentation	3855
6.851.2.1	URI	3855
6.851.2.2	URI	3855
6.851.2.3	URI	3856
6.851.2.4	URI	3856
6.851.2.5	URI	3856
6.851.2.6	URI	3856
6.851.2.7	URI	3857
6.851.2.8	~URI	3857
6.851.3	Member Function Documentation	3857
6.851.3.1	compareTo	3857
6.851.3.2	create	3857
6.851.3.3	equals	3858
6.851.3.4	getAuthority	3858
6.851.3.5	getFragment	3858
6.851.3.6	getHost	3858

6.851.3.7	getPath . . . . .	3858
6.851.3.8	getPort . . . . .	3858
6.851.3.9	getQuery . . . . .	3858
6.851.3.10	getRawAuthority . . . . .	3859
6.851.3.11	getRawFragment . . . . .	3859
6.851.3.12	getRawPath . . . . .	3859
6.851.3.13	getRawQuery . . . . .	3859
6.851.3.14	getRawSchemeSpecificPart . . . . .	3860
6.851.3.15	getRawUserInfo . . . . .	3860
6.851.3.16	getScheme . . . . .	3860
6.851.3.17	getSchemeSpecificPart . . . . .	3860
6.851.3.18	getUserInfo . . . . .	3860
6.851.3.19	isAbsolute . . . . .	3861
6.851.3.20	isOpaque . . . . .	3861
6.851.3.21	normalize . . . . .	3861
6.851.3.22	operator< . . . . .	3861
6.851.3.23	operator== . . . . .	3862
6.851.3.24	parseServerAuthority . . . . .	3862
6.851.3.25	relativize . . . . .	3863
6.851.3.26	resolve . . . . .	3863
6.851.3.27	resolve . . . . .	3863
6.851.3.28	toString . . . . .	3864
6.851.3.29	toURL . . . . .	3864
6.852	decaf::internal::net::URLEncoderDecoder Class Reference . . . . .	3865
6.852.1	Constructor & Destructor Documentation . . . . .	3866
6.852.1.1	URLEncoderDecoder . . . . .	3866
6.852.1.2	~URLEncoderDecoder . . . . .	3866
6.852.2	Member Function Documentation . . . . .	3866
6.852.2.1	decode . . . . .	3866
6.852.2.2	encodeOthers . . . . .	3866
6.852.2.3	quoteIllegal . . . . .	3866
6.852.2.4	validate . . . . .	3867
6.852.2.5	validateSimple . . . . .	3867
6.853	decaf::internal::net::URIHelper Class Reference . . . . .	3867

6.853.1 Detailed Description . . . . .	3869
6.853.2 Constructor & Destructor Documentation . . . . .	3869
6.853.2.1 URIHelper . . . . .	3869
6.853.2.2 URIHelper . . . . .	3869
6.853.2.3 ~URIHelper . . . . .	3869
6.853.3 Member Function Documentation . . . . .	3869
6.853.3.1 isValidDomainName . . . . .	3869
6.853.3.2 isValidHexChar . . . . .	3870
6.853.3.3 isValidHost . . . . .	3870
6.853.3.4 isValidIP4Word . . . . .	3870
6.853.3.5 isValidIP6Address . . . . .	3871
6.853.3.6 isValidIPv4Address . . . . .	3871
6.853.3.7 parseAuthority . . . . .	3871
6.853.3.8 parseURI . . . . .	3872
6.853.3.9 validateAuthority . . . . .	3872
6.853.3.10 validateFragment . . . . .	3872
6.853.3.11 validatePath . . . . .	3873
6.853.3.12 validateQuery . . . . .	3873
6.853.3.13 validateScheme . . . . .	3873
6.853.3.14 validateSsp . . . . .	3874
6.853.3.15 validateUserInfo . . . . .	3874
6.854 activemq::transport::failover::URIPool Class Reference . . . . .	3875
6.854.1 Constructor & Destructor Documentation . . . . .	3875
6.854.1.1 URIPool . . . . .	3875
6.854.1.2 URIPool . . . . .	3875
6.854.1.3 ~URIPool . . . . .	3876
6.854.2 Member Function Documentation . . . . .	3876
6.854.2.1 addURI . . . . .	3876
6.854.2.2 addURIs . . . . .	3876
6.854.2.3 getURI . . . . .	3876
6.854.2.4 isRandomize . . . . .	3876
6.854.2.5 removeURI . . . . .	3877
6.854.2.6 setRandomize . . . . .	3877
6.855 activemq::util::URISupport Class Reference . . . . .	3877

6.855.1 Member Function Documentation . . . . .	3878
6.855.1.1 createQueryString . . . . .	3878
6.855.1.2 parseComposite . . . . .	3878
6.855.1.3 parseQuery . . . . .	3879
6.855.1.4 parseQuery . . . . .	3879
6.855.1.5 parseURL . . . . .	3879
6.856decaf::net::URISyntaxException Class Reference . . . . .	3880
6.856.1 Constructor & Destructor Documentation . . . . .	3881
6.856.1.1 URISyntaxException . . . . .	3881
6.856.1.2 URISyntaxException . . . . .	3881
6.856.1.3 URISyntaxException . . . . .	3881
6.856.1.4 URISyntaxException . . . . .	3881
6.856.1.5 URISyntaxException . . . . .	3881
6.856.1.6 URISyntaxException . . . . .	3882
6.856.1.7 URISyntaxException . . . . .	3882
6.856.1.8 URISyntaxException . . . . .	3882
6.856.1.9 ~URISyntaxException . . . . .	3883
6.856.2 Member Function Documentation . . . . .	3883
6.856.2.1 clone . . . . .	3883
6.856.2.2 getIndex . . . . .	3883
6.856.2.3 getInput . . . . .	3883
6.856.2.4 getReason . . . . .	3883
6.857decaf::internal::net::URIType Class Reference . . . . .	3884
6.857.1 Detailed Description . . . . .	3885
6.857.2 Constructor & Destructor Documentation . . . . .	3885
6.857.2.1 URIType . . . . .	3885
6.857.2.2 URIType . . . . .	3885
6.857.2.3 ~URIType . . . . .	3885
6.857.3 Member Function Documentation . . . . .	3885
6.857.3.1 getAuthority . . . . .	3885
6.857.3.2 getFragment . . . . .	3886
6.857.3.3 getHost . . . . .	3886
6.857.3.4 getPath . . . . .	3886
6.857.3.5 getPort . . . . .	3886

6.857.3.6	getQuery . . . . .	3886
6.857.3.7	getScheme . . . . .	3887
6.857.3.8	getSchemeSpecificPart . . . . .	3887
6.857.3.9	getSource . . . . .	3887
6.857.3.10	getUserInfo . . . . .	3887
6.857.3.11	isAbsolute . . . . .	3887
6.857.3.12	isOpaque . . . . .	3888
6.857.3.13	isServerAuthority . . . . .	3888
6.857.3.14	isValid . . . . .	3888
6.857.3.15	setAbsolute . . . . .	3888
6.857.3.16	setAuthority . . . . .	3888
6.857.3.17	setFragment . . . . .	3889
6.857.3.18	setHost . . . . .	3889
6.857.3.19	setOpaque . . . . .	3889
6.857.3.20	setPath . . . . .	3889
6.857.3.21	setPort . . . . .	3889
6.857.3.22	setQuery . . . . .	3890
6.857.3.23	setScheme . . . . .	3890
6.857.3.24	setSchemeSpecificPart . . . . .	3890
6.857.3.25	setServerAuthority . . . . .	3890
6.857.3.26	setSource . . . . .	3890
6.857.3.27	setUserInfo . . . . .	3891
6.857.3.28	setValid . . . . .	3891
6.858	decaf::net::URL Class Reference . . . . .	3891
6.858.1	Detailed Description . . . . .	3891
6.858.2	Constructor & Destructor Documentation . . . . .	3893
6.858.2.1	URL . . . . .	3893
6.858.2.2	URL . . . . .	3893
6.858.2.3	~URL . . . . .	3893
6.859	decaf::net::URLDecoder Class Reference . . . . .	3893
6.859.1	Constructor & Destructor Documentation . . . . .	3894
6.859.1.1	~URLDecoder . . . . .	3894
6.859.2	Member Function Documentation . . . . .	3894
6.859.2.1	decode . . . . .	3894

6.860	decaf::net::URLEncoder Class Reference . . . . .	3894
6.860.1	Constructor & Destructor Documentation . . . . .	3894
6.860.1.1	~URLEncoder . . . . .	3894
6.860.2	Member Function Documentation . . . . .	3895
6.860.2.1	encode . . . . .	3895
6.861	activemq::util::Usage Class Reference . . . . .	3895
6.861.1	Constructor & Destructor Documentation . . . . .	3896
6.861.1.1	~Usage . . . . .	3896
6.861.2	Member Function Documentation . . . . .	3896
6.861.2.1	decreaseUsage . . . . .	3896
6.861.2.2	enqueueUsage . . . . .	3896
6.861.2.3	increaseUsage . . . . .	3896
6.861.2.4	isFull . . . . .	3896
6.861.2.5	waitForSpace . . . . .	3897
6.861.2.6	waitForSpace . . . . .	3897
6.862	decaf::io::UTFDataFormatException Class Reference . . . . .	3897
6.862.1	Detailed Description . . . . .	3898
6.862.2	Constructor & Destructor Documentation . . . . .	3898
6.862.2.1	UTFDataFormatException . . . . .	3898
6.862.2.2	UTFDataFormatException . . . . .	3898
6.862.2.3	UTFDataFormatException . . . . .	3898
6.862.2.4	UTFDataFormatException . . . . .	3899
6.862.2.5	UTFDataFormatException . . . . .	3899
6.862.2.6	UTFDataFormatException . . . . .	3899
6.862.2.7	~UTFDataFormatException . . . . .	3899
6.862.3	Member Function Documentation . . . . .	3900
6.862.3.1	clone . . . . .	3900
6.863	decaf::util::UUID Class Reference . . . . .	3900
6.863.1	Detailed Description . . . . .	3901
6.863.2	Constructor & Destructor Documentation . . . . .	3902
6.863.2.1	UUID . . . . .	3902
6.863.2.2	~UUID . . . . .	3902
6.863.3	Member Function Documentation . . . . .	3902
6.863.3.1	clockSequence . . . . .	3902

6.863.3.2 compareTo . . . . .	3903
6.863.3.3 equals . . . . .	3903
6.863.3.4 fromString . . . . .	3903
6.863.3.5 getLeastSignificantBits . . . . .	3903
6.863.3.6 getMostSignificantBits . . . . .	3903
6.863.3.7 nameUUIDFromBytes . . . . .	3904
6.863.3.8 nameUUIDFromBytes . . . . .	3904
6.863.3.9 node . . . . .	3904
6.863.3.10 operator< . . . . .	3905
6.863.3.11 operator== . . . . .	3905
6.863.3.12 randomUUID . . . . .	3905
6.863.3.13 timestamp . . . . .	3905
6.863.3.14 toString . . . . .	3906
6.863.3.15 variant . . . . .	3906
6.863.3.16 version . . . . .	3906
6.864activemq::wireformat::WireFormat Class Reference . . . . .	3907
6.864.1 Detailed Description . . . . .	3908
6.864.2 Constructor & Destructor Documentation . . . . .	3908
6.864.2.1 ~WireFormat . . . . .	3908
6.864.3 Member Function Documentation . . . . .	3908
6.864.3.1 createNegotiator . . . . .	3908
6.864.3.2 getVersion . . . . .	3909
6.864.3.3 hasNegotiator . . . . .	3909
6.864.3.4 inReceive . . . . .	3909
6.864.3.5 marshal . . . . .	3910
6.864.3.6 setVersion . . . . .	3910
6.864.3.7 unmarshal . . . . .	3910
6.865activemq::wireformat::WireFormatFactory Class Reference . . . . .	3911
6.865.1 Detailed Description . . . . .	3911
6.865.2 Constructor & Destructor Documentation . . . . .	3911
6.865.2.1 ~WireFormatFactory . . . . .	3911
6.865.3 Member Function Documentation . . . . .	3912
6.865.3.1 createWireFormat . . . . .	3912
6.866activemq::commands::WireFormatInfo Class Reference . . . . .	3912



6.866.1 Constructor & Destructor Documentation . . . . .	3914
6.866.1.1 WireFormatInfo . . . . .	3914
6.866.1.2 ~WireFormatInfo . . . . .	3914
6.866.2 Member Function Documentation . . . . .	3914
6.866.2.1 afterUnmarshal . . . . .	3914
6.866.2.2 beforeMarshal . . . . .	3915
6.866.2.3 cloneDataStructure . . . . .	3915
6.866.2.4 copyDataStructure . . . . .	3915
6.866.2.5 equals . . . . .	3915
6.866.2.6 getCacheSize . . . . .	3916
6.866.2.7 getDataStructureType . . . . .	3916
6.866.2.8 getMagic . . . . .	3916
6.866.2.9 getMarshaledProperties . . . . .	3916
6.866.2.10getMaxInactivityDuration . . . . .	3917
6.866.2.11getMaxInactivityDurationInitalDelay . . . . .	3917
6.866.2.12getProperties . . . . .	3917
6.866.2.13getProperties . . . . .	3917
6.866.2.14getVersion . . . . .	3917
6.866.2.15sCacheEnabled . . . . .	3918
6.866.2.16sMarshalAware . . . . .	3918
6.866.2.17sSizePrefixDisabled . . . . .	3918
6.866.2.18sStackTraceEnabled . . . . .	3918
6.866.2.19sTcpNoDelayEnabled . . . . .	3918
6.866.2.20sTightEncodingEnabled . . . . .	3919
6.866.2.21isValid . . . . .	3919
6.866.2.22sWireFormatInfo . . . . .	3919
6.866.2.23setCacheEnabled . . . . .	3919
6.866.2.24setCacheSize . . . . .	3919
6.866.2.25setMagic . . . . .	3920
6.866.2.26setMarshaledProperties . . . . .	3920
6.866.2.27setMaxInactivityDuration . . . . .	3920
6.866.2.28setMaxInactivityDurationInitalDelay . . . . .	3920
6.866.2.29setProperties . . . . .	3920
6.866.2.30setSizePrefixDisabled . . . . .	3921

6.866.2.31	setStackTraceEnabled . . . . .	3921
6.866.2.32	setTcpNoDelayEnabled . . . . .	3921
6.866.2.33	setTightEncodingEnabled . . . . .	3921
6.866.2.34	setVersion . . . . .	3922
6.866.2.35	toString . . . . .	3922
6.866.2.36	visit . . . . .	3922
6.866.3	Field Documentation . . . . .	3922
6.866.3.1	ID_WIREFORMATINFO . . . . .	3922
6.867	activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	
	Class Reference . . . . .	3923
6.867.1	Detailed Description . . . . .	3923
6.867.2	Constructor & Destructor Documentation . . . . .	3924
6.867.2.1	WireFormatInfoMarshaller . . . . .	3924
6.867.2.2	~WireFormatInfoMarshaller . . . . .	3924
6.867.3	Member Function Documentation . . . . .	3924
6.867.3.1	createObject . . . . .	3924
6.867.3.2	getDataStructureType . . . . .	3924
6.867.3.3	looseMarshal . . . . .	3924
6.867.3.4	looseUnmarshal . . . . .	3925
6.867.3.5	tightMarshal1 . . . . .	3925
6.867.3.6	tightMarshal2 . . . . .	3926
6.867.3.7	tightUnmarshal . . . . .	3926
6.868	activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller	
	Class Reference . . . . .	3927
6.868.1	Detailed Description . . . . .	3927
6.868.2	Constructor & Destructor Documentation . . . . .	3928
6.868.2.1	WireFormatInfoMarshaller . . . . .	3928
6.868.2.2	~WireFormatInfoMarshaller . . . . .	3928
6.868.3	Member Function Documentation . . . . .	3928
6.868.3.1	createObject . . . . .	3928
6.868.3.2	getDataStructureType . . . . .	3928
6.868.3.3	looseMarshal . . . . .	3928
6.868.3.4	looseUnmarshal . . . . .	3929
6.868.3.5	tightMarshal1 . . . . .	3929

6.868.3.6	tightMarshal2 . . . . .	3930
6.868.3.7	tightUnmarshal . . . . .	3930
6.869	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	
	Class Reference . . . . .	3931
6.869.1	Detailed Description . . . . .	3931
6.869.2	Constructor & Destructor Documentation . . . . .	3932
6.869.2.1	WireFormatInfoMarshaller . . . . .	3932
6.869.2.2	~WireFormatInfoMarshaller . . . . .	3932
6.869.3	Member Function Documentation . . . . .	3932
6.869.3.1	createObject . . . . .	3932
6.869.3.2	getDataStructureType . . . . .	3932
6.869.3.3	looseMarshal . . . . .	3932
6.869.3.4	looseUnmarshal . . . . .	3933
6.869.3.5	tightMarshal1 . . . . .	3933
6.869.3.6	tightMarshal2 . . . . .	3934
6.869.3.7	tightUnmarshal . . . . .	3934
6.870	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	
	Class Reference . . . . .	3935
6.870.1	Detailed Description . . . . .	3935
6.870.2	Constructor & Destructor Documentation . . . . .	3936
6.870.2.1	WireFormatInfoMarshaller . . . . .	3936
6.870.2.2	~WireFormatInfoMarshaller . . . . .	3936
6.870.3	Member Function Documentation . . . . .	3936
6.870.3.1	createObject . . . . .	3936
6.870.3.2	getDataStructureType . . . . .	3936
6.870.3.3	looseMarshal . . . . .	3936
6.870.3.4	looseUnmarshal . . . . .	3937
6.870.3.5	tightMarshal1 . . . . .	3937
6.870.3.6	tightMarshal2 . . . . .	3938
6.870.3.7	tightUnmarshal . . . . .	3938
6.871	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	
	Class Reference . . . . .	3939
6.871.1	Detailed Description . . . . .	3939
6.871.2	Constructor & Destructor Documentation . . . . .	3940
6.871.2.1	WireFormatInfoMarshaller . . . . .	3940

6.871.2.2 ~WireFormatInfoMarshaller . . . . .	3940
6.871.3 Member Function Documentation . . . . .	3940
6.871.3.1 createObject . . . . .	3940
6.871.3.2 getDataStructureType . . . . .	3940
6.871.3.3 looseMarshal . . . . .	3940
6.871.3.4 looseUnmarshal . . . . .	3941
6.871.3.5 tightMarshal1 . . . . .	3941
6.871.3.6 tightMarshal2 . . . . .	3942
6.871.3.7 tightUnmarshal . . . . .	3942
6.872activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	
Class Reference . . . . .	3943
6.872.1 Detailed Description . . . . .	3943
6.872.2 Constructor & Destructor Documentation . . . . .	3944
6.872.2.1 WireFormatInfoMarshaller . . . . .	3944
6.872.2.2 ~WireFormatInfoMarshaller . . . . .	3944
6.872.3 Member Function Documentation . . . . .	3944
6.872.3.1 createObject . . . . .	3944
6.872.3.2 getDataStructureType . . . . .	3944
6.872.3.3 looseMarshal . . . . .	3944
6.872.3.4 looseUnmarshal . . . . .	3945
6.872.3.5 tightMarshal1 . . . . .	3945
6.872.3.6 tightMarshal2 . . . . .	3946
6.872.3.7 tightUnmarshal . . . . .	3946
6.873activemq::wireformat::WireFormatNegotiator Class Reference . . . . .	3946
6.873.1 Detailed Description . . . . .	3947
6.873.2 Constructor & Destructor Documentation . . . . .	3947
6.873.2.1 WireFormatNegotiator . . . . .	3947
6.873.2.2 ~WireFormatNegotiator . . . . .	3947
6.874activemq::wireformat::WireFormatRegistry Class Reference . . . . .	3947
6.874.1 Detailed Description . . . . .	3948
6.874.2 Constructor & Destructor Documentation . . . . .	3948
6.874.2.1 ~WireFormatRegistry . . . . .	3948
6.874.3 Member Function Documentation . . . . .	3948
6.874.3.1 findFactory . . . . .	3948

6.874.3.2	getInstance . . . . .	3949
6.874.3.3	getWireFormatNames . . . . .	3949
6.874.3.4	registerFactory . . . . .	3949
6.874.3.5	unregisterFactory . . . . .	3950
6.875	activemq::transport::inactivity::WriteChecker Class Reference . . . . .	3950
6.875.1	Detailed Description . . . . .	3950
6.875.2	Constructor & Destructor Documentation . . . . .	3951
6.875.2.1	WriteChecker . . . . .	3951
6.875.2.2	~WriteChecker . . . . .	3951
6.875.3	Member Function Documentation . . . . .	3951
6.875.3.1	run . . . . .	3951
6.876	decaf::io::Writer Class Reference . . . . .	3951
6.876.1	Constructor & Destructor Documentation . . . . .	3952
6.876.1.1	Writer . . . . .	3952
6.876.1.2	~Writer . . . . .	3952
6.876.2	Member Function Documentation . . . . .	3952
6.876.2.1	append . . . . .	3953
6.876.2.2	append . . . . .	3953
6.876.2.3	append . . . . .	3953
6.876.2.4	doAppendChar . . . . .	3954
6.876.2.5	doAppendCharSequence . . . . .	3954
6.876.2.6	doAppendCharSequenceStartEnd . . . . .	3954
6.876.2.7	doWriteArray . . . . .	3954
6.876.2.8	doWriteArrayBounded . . . . .	3954
6.876.2.9	doWriteChar . . . . .	3955
6.876.2.10	doWriteString . . . . .	3955
6.876.2.11	doWriteStringBounded . . . . .	3955
6.876.2.12	doWriteVector . . . . .	3955
6.876.2.13	write . . . . .	3955
6.876.2.14	write . . . . .	3955
6.876.2.15	write . . . . .	3955
6.876.2.16	write . . . . .	3956
6.876.2.17	write . . . . .	3956
6.876.2.18	write . . . . .	3956

6.877decaf::security::auth::x500::X500Principal Class Reference . . . . .	3957
6.877.1 Constructor & Destructor Documentation . . . . .	3957
6.877.1.1 ~X500Principal . . . . .	3957
6.877.2 Member Function Documentation . . . . .	3957
6.877.2.1 getEncoded . . . . .	3957
6.877.2.2 getName . . . . .	3957
6.877.2.3 hashCode . . . . .	3958
6.878decaf::security::cert::X509Certificate Class Reference . . . . .	3958
6.878.1 Detailed Description . . . . .	3959
6.878.2 Constructor & Destructor Documentation . . . . .	3959
6.878.2.1 ~X509Certificate . . . . .	3959
6.878.3 Member Function Documentation . . . . .	3959
6.878.3.1 checkValidity . . . . .	3959
6.878.3.2 checkValidity . . . . .	3959
6.878.3.3 getBasicConstraints . . . . .	3959
6.878.3.4 getIssuerUniqueID . . . . .	3959
6.878.3.5 getIssuerX500Principal . . . . .	3959
6.878.3.6 getKeyUsage . . . . .	3959
6.878.3.7 getNotAfter . . . . .	3959
6.878.3.8 getNotBefore . . . . .	3959
6.878.3.9 getSigAlgName . . . . .	3959
6.878.3.10getSigAlgOID . . . . .	3959
6.878.3.11getSigAlgParams . . . . .	3959
6.878.3.12getSignature . . . . .	3960
6.878.3.13getSubjectUniqueID . . . . .	3960
6.878.3.14getSubjectX500Principal . . . . .	3960
6.878.3.15getTBSCertificate . . . . .	3960
6.878.3.16getVersion . . . . .	3960
6.879activemq::commands::XATransactionId Class Reference . . . . .	3960
6.879.1 Member Typedef Documentation . . . . .	3961
6.879.1.1 COMPARATOR . . . . .	3961
6.879.2 Constructor & Destructor Documentation . . . . .	3961
6.879.2.1 XATransactionId . . . . .	3961
6.879.2.2 XATransactionId . . . . .	3962

6.879.2.3 ~XATransactionId . . . . .	3962
6.879.3 Member Function Documentation . . . . .	3962
6.879.3.1 cloneDataStructure . . . . .	3962
6.879.3.2 compareTo . . . . .	3962
6.879.3.3 copyDataStructure . . . . .	3962
6.879.3.4 equals . . . . .	3962
6.879.3.5 equals . . . . .	3963
6.879.3.6 getBranchQualifier . . . . .	3963
6.879.3.7 getBranchQualifier . . . . .	3963
6.879.3.8 getDataStructureType . . . . .	3963
6.879.3.9 getFormatId . . . . .	3963
6.879.3.10getGlobalTransactionId . . . . .	3963
6.879.3.11getGlobalTransactionId . . . . .	3963
6.879.3.12operator< . . . . .	3963
6.879.3.13operator= . . . . .	3963
6.879.3.14operator== . . . . .	3963
6.879.3.15setBranchQualifier . . . . .	3963
6.879.3.16setFormatId . . . . .	3963
6.879.3.17setGlobalTransactionId . . . . .	3964
6.879.3.18toString . . . . .	3964
6.879.4 Field Documentation . . . . .	3964
6.879.4.1 branchQualifier . . . . .	3964
6.879.4.2 formatId . . . . .	3964
6.879.4.3 globalTransactionId . . . . .	3964
6.879.4.4 ID_XATRANSACTIONID . . . . .	3964
6.880activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller	
Class Reference . . . . .	3964
6.880.1 Detailed Description . . . . .	3965
6.880.2 Constructor & Destructor Documentation . . . . .	3965
6.880.2.1 XATransactionIdMarshaller . . . . .	3965
6.880.2.2 ~XATransactionIdMarshaller . . . . .	3965
6.880.3 Member Function Documentation . . . . .	3965
6.880.3.1 createObject . . . . .	3965
6.880.3.2 getDataStructureType . . . . .	3966

6.880.3.3 looseMarshal . . . . .	3966
6.880.3.4 looseUnmarshal . . . . .	3966
6.880.3.5 tightMarshal1 . . . . .	3967
6.880.3.6 tightMarshal2 . . . . .	3967
6.880.3.7 tightUnmarshal . . . . .	3968
6.881activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	
Class Reference . . . . .	3968
6.881.1 Detailed Description . . . . .	3969
6.881.2 Constructor & Destructor Documentation . . . . .	3969
6.881.2.1 XATransactionIdMarshaller . . . . .	3969
6.881.2.2 ~XATransactionIdMarshaller . . . . .	3969
6.881.3 Member Function Documentation . . . . .	3969
6.881.3.1 createObject . . . . .	3969
6.881.3.2 getDataStructureType . . . . .	3970
6.881.3.3 looseMarshal . . . . .	3970
6.881.3.4 looseUnmarshal . . . . .	3970
6.881.3.5 tightMarshal1 . . . . .	3971
6.881.3.6 tightMarshal2 . . . . .	3971
6.881.3.7 tightUnmarshal . . . . .	3972
6.882activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	
Class Reference . . . . .	3972
6.882.1 Detailed Description . . . . .	3973
6.882.2 Constructor & Destructor Documentation . . . . .	3973
6.882.2.1 XATransactionIdMarshaller . . . . .	3973
6.882.2.2 ~XATransactionIdMarshaller . . . . .	3973
6.882.3 Member Function Documentation . . . . .	3973
6.882.3.1 createObject . . . . .	3973
6.882.3.2 getDataStructureType . . . . .	3974
6.882.3.3 looseMarshal . . . . .	3974
6.882.3.4 looseUnmarshal . . . . .	3974
6.882.3.5 tightMarshal1 . . . . .	3975
6.882.3.6 tightMarshal2 . . . . .	3975
6.882.3.7 tightUnmarshal . . . . .	3976
6.883activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	
Class Reference . . . . .	3976



6.883.1 Detailed Description . . . . .	3977
6.883.2 Constructor & Destructor Documentation . . . . .	3977
6.883.2.1 XATransactionIdMarshaller . . . . .	3977
6.883.2.2 ~XATransactionIdMarshaller . . . . .	3977
6.883.3 Member Function Documentation . . . . .	3977
6.883.3.1 createObject . . . . .	3977
6.883.3.2 getDataStructureType . . . . .	3978
6.883.3.3 looseMarshal . . . . .	3978
6.883.3.4 looseUnmarshal . . . . .	3978
6.883.3.5 tightMarshal1 . . . . .	3979
6.883.3.6 tightMarshal2 . . . . .	3979
6.883.3.7 tightUnmarshal . . . . .	3980
6.884activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	
Class Reference . . . . .	3980
6.884.1 Detailed Description . . . . .	3981
6.884.2 Constructor & Destructor Documentation . . . . .	3981
6.884.2.1 XATransactionIdMarshaller . . . . .	3981
6.884.2.2 ~XATransactionIdMarshaller . . . . .	3981
6.884.3 Member Function Documentation . . . . .	3981
6.884.3.1 createObject . . . . .	3981
6.884.3.2 getDataStructureType . . . . .	3982
6.884.3.3 looseMarshal . . . . .	3982
6.884.3.4 looseUnmarshal . . . . .	3982
6.884.3.5 tightMarshal1 . . . . .	3983
6.884.3.6 tightMarshal2 . . . . .	3983
6.884.3.7 tightUnmarshal . . . . .	3984
6.885activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	
Class Reference . . . . .	3984
6.885.1 Detailed Description . . . . .	3985
6.885.2 Constructor & Destructor Documentation . . . . .	3985
6.885.2.1 XATransactionIdMarshaller . . . . .	3985
6.885.2.2 ~XATransactionIdMarshaller . . . . .	3985
6.885.3 Member Function Documentation . . . . .	3985
6.885.3.1 createObject . . . . .	3985

6.885.3.2	getDataStructureType . . . . .	3986
6.885.3.3	looseMarshal . . . . .	3986
6.885.3.4	looseUnmarshal . . . . .	3986
6.885.3.5	tightMarshal1 . . . . .	3987
6.885.3.6	tightMarshal2 . . . . .	3987
6.885.3.7	tightUnmarshal . . . . .	3988
6.886	decaf::util::logging::XMLFormatter Class Reference . . . . .	3988
6.886.1	Detailed Description . . . . .	3989
6.886.2	Constructor & Destructor Documentation . . . . .	3989
6.886.2.1	XMLFormatter . . . . .	3989
6.886.2.2	~XMLFormatter . . . . .	3989
6.886.3	Member Function Documentation . . . . .	3989
6.886.3.1	format . . . . .	3989
6.886.3.2	getHead . . . . .	3989
6.886.3.3	getTail . . . . .	3990
6.887	z_stream_s Struct Reference . . . . .	3990
6.887.1	Field Documentation . . . . .	3991
6.887.1.1	adler . . . . .	3991
6.887.1.2	avail_in . . . . .	3991
6.887.1.3	avail_out . . . . .	3991
6.887.1.4	data_type . . . . .	3991
6.887.1.5	msg . . . . .	3991
6.887.1.6	next_in . . . . .	3991
6.887.1.7	next_out . . . . .	3991
6.887.1.8	opaque . . . . .	3991
6.887.1.9	reserved . . . . .	3991
6.887.1.10	state . . . . .	3991
6.887.1.11	total_in . . . . .	3991
6.887.1.12	total_out . . . . .	3991
6.887.1.13	zalloc . . . . .	3991
6.887.1.14	zfree . . . . .	3991
6.888	decaf::util::zip::ZipException Class Reference . . . . .	3991
6.888.1	Constructor & Destructor Documentation . . . . .	3992
6.888.1.1	ZipException . . . . .	3992

6.888.1.2	ZipException . . . . .	3992
6.888.1.3	ZipException . . . . .	3992
6.888.1.4	ZipException . . . . .	3992
6.888.1.5	ZipException . . . . .	3993
6.888.1.6	ZipException . . . . .	3993
6.888.1.7	~ZipException . . . . .	3993
6.888.2	Member Function Documentation . . . . .	3993
6.888.2.1	clone . . . . .	3993
<b>7</b>	<b>File Documentation</b>	<b>3995</b>
7.1	src/main/activemq/cmsutil/CachedConsumer.h File Reference . . . . .	3995
7.2	src/main/activemq/cmsutil/CachedProducer.h File Reference . . . . .	3995
7.3	src/main/activemq/cmsutil/CmsAccessor.h File Reference . . . . .	3996
7.4	src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference . . . . .	3996
7.5	src/main/activemq/cmsutil/CmsTemplate.h File Reference . . . . .	3997
7.6	src/main/activemq/cmsutil/DestinationResolver.h File Reference . . . . .	3997
7.7	src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference . . . . .	3998
7.8	src/main/activemq/cmsutil/MessageCreator.h File Reference . . . . .	3998
7.9	src/main/activemq/cmsutil/PooledSession.h File Reference . . . . .	3999
7.10	src/main/activemq/cmsutil/ProducerCallback.h File Reference . . . . .	3999
7.11	src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference . . . . .	4000
7.12	src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference . . . . .	4000
7.13	src/main/activemq/cmsutil/SessionCallback.h File Reference . . . . .	4001
7.14	src/main/activemq/cmsutil/SessionPool.h File Reference . . . . .	4001
7.15	src/main/activemq/commands/ActiveMQBlobMessage.h File Reference . . . . .	4002
7.16	src/main/activemq/commands/ActiveMQBytesMessage.h File Reference . . . . .	4002
7.17	src/main/activemq/commands/ActiveMQDestination.h File Reference . . . . .	4003
7.18	src/main/activemq/commands/ActiveMQMapMessage.h File Reference . . . . .	4004
7.19	src/main/activemq/commands/ActiveMQMessage.h File Reference . . . . .	4004
7.20	src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference . . . . .	4005
7.21	src/main/activemq/commands/ActiveMQObjectMessage.h File Reference . . . . .	4006
7.22	src/main/activemq/commands/ActiveMQQueue.h File Reference . . . . .	4006
7.23	src/main/activemq/commands/ActiveMQStreamMessage.h File Reference . . . . .	4007

7.24	src/main/activemq/commands/ActiveMQTempDestination.h File Reference	4007
7.25	src/main/activemq/commands/ActiveMQTempQueue.h File Reference	4008
7.26	src/main/activemq/commands/ActiveMQTempTopic.h File Reference	4009
7.27	src/main/activemq/commands/ActiveMQTextMessage.h File Reference	4009
7.28	src/main/activemq/commands/ActiveMQTopic.h File Reference	4010
7.29	src/main/activemq/commands/BaseCommand.h File Reference	4010
7.30	src/main/activemq/commands/BaseDataStructure.h File Reference	4011
7.31	src/main/activemq/commands/BooleanExpression.h File Reference	4011
7.32	src/main/activemq/commands/BrokerError.h File Reference	4012
7.33	src/main/activemq/commands/BrokerId.h File Reference	4012
7.34	src/main/activemq/commands/BrokerInfo.h File Reference	4013
7.35	src/main/activemq/commands/Command.h File Reference	4013
7.36	src/main/activemq/commands/ConnectionControl.h File Reference	4014
7.37	src/main/activemq/commands/ConnectionError.h File Reference	4014
7.38	src/main/activemq/commands/ConnectionId.h File Reference	4015
7.39	src/main/activemq/commands/ConnectionInfo.h File Reference	4015
7.40	src/main/activemq/commands/ConsumerControl.h File Reference	4016
7.41	src/main/activemq/commands/ConsumerId.h File Reference	4016
7.42	src/main/activemq/commands/ConsumerInfo.h File Reference	4017
7.43	src/main/activemq/commands/ControlCommand.h File Reference	4018
7.44	src/main/activemq/commands/DataArrayResponse.h File Reference	4018
7.45	src/main/activemq/commands/DataResponse.h File Reference	4019
7.46	src/main/activemq/commands/DataStructure.h File Reference	4019
7.47	src/main/activemq/commands/DestinationInfo.h File Reference	4020
7.48	src/main/activemq/commands/DiscoveryEvent.h File Reference	4020
7.49	src/main/activemq/commands/ExceptionResponse.h File Reference	4021
7.50	src/main/activemq/commands/FlushCommand.h File Reference	4021
7.51	src/main/activemq/commands/IntegerResponse.h File Reference	4022
7.52	src/main/activemq/commands/JournalQueueAck.h File Reference	4022
7.53	src/main/activemq/commands/JournalTopicAck.h File Reference	4023
7.54	src/main/activemq/commands/JournalTrace.h File Reference	4023
7.55	src/main/activemq/commands/JournalTransaction.h File Reference	4024
7.56	src/main/activemq/commands/KeepAliveInfo.h File Reference	4024
7.57	src/main/activemq/commands/LastPartialCommand.h File Reference	4025

7.58	src/main/activemq/commands/LocalTransactionId.h File Reference . . .	4025
7.59	src/main/activemq/commands/Message.h File Reference . . . . .	4026
7.60	src/main/cms/Message.h File Reference . . . . .	4027
7.61	src/main/activemq/commands/MessageAck.h File Reference . . . . .	4027
7.62	src/main/activemq/commands/MessageDispatch.h File Reference . . . .	4028
7.63	src/main/activemq/commands/MessageDispatchNotification.h File Reference . . . . .	4029
7.64	src/main/activemq/commands/Messageld.h File Reference . . . . .	4029
7.65	src/main/activemq/commands/MessagePull.h File Reference . . . . .	4030
7.66	src/main/activemq/commands/NetworkBridgeFilter.h File Reference . . .	4030
7.67	src/main/activemq/commands/PartialCommand.h File Reference . . . .	4031
7.68	src/main/activemq/commands/ProducerAck.h File Reference . . . . .	4031
7.69	src/main/activemq/commands/ProducerId.h File Reference . . . . .	4032
7.70	src/main/activemq/commands/ProducerInfo.h File Reference . . . . .	4032
7.71	src/main/activemq/commands/RemoveInfo.h File Reference . . . . .	4033
7.72	src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference	4034
7.73	src/main/activemq/commands/ReplayCommand.h File Reference . . . .	4034
7.74	src/main/activemq/commands/Response.h File Reference . . . . .	4035
7.75	src/main/activemq/commands/SessionId.h File Reference . . . . .	4035
7.76	src/main/activemq/commands/SessionInfo.h File Reference . . . . .	4036
7.77	src/main/activemq/commands/ShutdownInfo.h File Reference . . . . .	4036
7.78	src/main/activemq/commands/SubscriptionInfo.h File Reference . . . .	4037
7.79	src/main/activemq/commands/TransactionId.h File Reference . . . . .	4037
7.80	src/main/activemq/commands/TransactionInfo.h File Reference . . . . .	4038
7.81	src/main/activemq/commands/WireFormatInfo.h File Reference . . . . .	4038
7.82	src/main/activemq/commands/XATransactionId.h File Reference . . . . .	4039
7.83	src/main/activemq/core/ActiveMQAckHandler.h File Reference . . . . .	4039
7.84	src/main/activemq/core/ActiveMQConnection.h File Reference . . . . .	4040
7.85	src/main/activemq/core/ActiveMQConnectionFactory.h File Reference . .	4040
7.86	src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference	4041
7.87	src/main/activemq/core/ActiveMQConstants.h File Reference . . . . .	4041
7.88	src/main/activemq/core/ActiveMQConsumer.h File Reference . . . . .	4042
7.89	src/main/activemq/core/ActiveMQProducer.h File Reference . . . . .	4043
7.90	src/main/activemq/core/ActiveMQQueueBrowser.h File Reference . . . .	4043

7.91	src/main/activemq/core/ActiveMQSession.h File Reference . . . . .	4044
7.92	src/main/activemq/core/ActiveMQSessionExecutor.h File Reference . . .	4045
7.93	src/main/activemq/core/ActiveMQTransactionContext.h File Reference .	4046
7.94	src/main/activemq/core/DispatchData.h File Reference . . . . .	4046
7.95	src/main/activemq/core/Dispatcher.h File Reference . . . . .	4047
7.96	src/main/activemq/core/MessageDispatchChannel.h File Reference . . .	4047
7.97	src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference .	4048
7.98	src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference	4048
7.99	src/main/activemq/core/PrefetchPolicy.h File Reference . . . . .	4049
7.100	src/main/activemq/core/RedeliveryPolicy.h File Reference . . . . .	4049
7.101	src/main/activemq/core/Synchronization.h File Reference . . . . .	4050
7.102	src/main/activemq/exceptions/ActiveMQException.h File Reference . . .	4050
7.103	src/main/activemq/exceptions/BrokerException.h File Reference . . . .	4051
7.104	src/main/activemq/exceptions/ExceptionDefines.h File Reference . . . .	4051
7.104.1	Define Documentation . . . . .	4052
7.104.1.1	AMQ_CATCH_EXCEPTION_CONVERT . . . . .	4052
7.104.1.2	AMQ_CATCH_NOTHROW . . . . .	4052
7.104.1.3	AMQ_CATCH_RETHROW . . . . .	4052
7.104.1.4	AMQ_CATCHALL_NOTHROW . . . . .	4053
7.104.1.5	AMQ_CATCHALL_THROW . . . . .	4053
7.105	src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference . . .	4053
7.105.1	Define Documentation . . . . .	4054
7.105.1.1	DECAF_CATCH_EXCEPTION_CONVERT . . . . .	4054
7.105.1.2	DECAF_CATCH_NOTHROW . . . . .	4054
7.105.1.3	DECAF_CATCH_RETHROW . . . . .	4054
7.105.1.4	DECAF_CATCHALL_NOTHROW . . . . .	4055
7.105.1.5	DECAF_CATCHALL_THROW . . . . .	4055
7.106	src/main/activemq/io/LoggingInputStream.h File Reference . . . . .	4055
7.107	src/main/activemq/io/LoggingOutputStream.h File Reference . . . . .	4056
7.108	src/main/activemq/library/ActiveMQCPP.h File Reference . . . . .	4056
7.109	src/main/activemq/state/CommandVisitor.h File Reference . . . . .	4057
7.110	src/main/activemq/state/CommandVisitorAdapter.h File Reference . . .	4057
7.111	src/main/activemq/state/ConnectionState.h File Reference . . . . .	4059
7.112	src/main/activemq/state/ConnectionStateTracker.h File Reference . . . .	4059

7.113src/main/activemq/state/ConsumerState.h File Reference . . . . .	4060
7.114src/main/activemq/state/ProducerState.h File Reference . . . . .	4061
7.115src/main/activemq/state/SessionState.h File Reference . . . . .	4061
7.116src/main/activemq/state/Tracked.h File Reference . . . . .	4062
7.117src/main/activemq/state/TransactionState.h File Reference . . . . .	4062
7.118src/main/activemq/threads/CompositeTask.h File Reference . . . . .	4063
7.119src/main/activemq/threads/CompositeTaskRunner.h File Reference . . .	4063
7.120src/main/activemq/threads/DedicatedTaskRunner.h File Reference . . .	4064
7.121src/main/activemq/threads/Task.h File Reference . . . . .	4065
7.122src/main/activemq/threads/TaskRunner.h File Reference . . . . .	4065
7.123src/main/activemq/transport/AbstractTransportFactory.h File Reference .	4065
7.124src/main/activemq/transport/CompositeTransport.h File Reference . . .	4066
7.125src/main/activemq/transport/correlator/FutureResponse.h File Reference	4067
7.126src/main/activemq/transport/correlator/ResponseCorrelator.h File Refer- ence . . . . .	4067
7.127src/main/activemq/transport/DefaultTransportListener.h File Reference .	4068
7.128src/main/activemq/transport/failover/BackupTransport.h File Reference .	4068
7.129src/main/activemq/transport/failover/BackupTransportPool.h File Refer- ence . . . . .	4069
7.130src/main/activemq/transport/failover/CloseTransportsTask.h File Reference	4070
7.131src/main/activemq/transport/failover/FailoverTransport.h File Reference .	4070
7.132src/main/activemq/transport/failover/FailoverTransportFactory.h File Ref- erence . . . . .	4071
7.133src/main/activemq/transport/failover/FailoverTransportListener.h File Ref- erence . . . . .	4072
7.134src/main/activemq/transport/failover/URIPool.h File Reference . . . . .	4072
7.135src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference .	4073
7.136src/main/activemq/transport/inactivity/ReadChecker.h File Reference . .	4073
7.137src/main/activemq/transport/inactivity/WriteChecker.h File Reference . .	4074
7.138src/main/activemq/transport/IOTransport.h File Reference . . . . .	4074
7.139src/main/activemq/transport/logging/LoggingTransport.h File Reference .	4075
7.140src/main/activemq/transport/mock/InternalCommandListener.h File Ref- erence . . . . .	4076
7.141src/main/activemq/transport/mock/MockTransport.h File Reference . . .	4076
7.142src/main/activemq/transport/mock/MockTransportFactory.h File Reference	4077

7.143src/main/activemq/transport/mock/ResponseBuilder.h File Reference . . .	4078
7.144src/main/activemq/transport/tcp/SslTransport.h File Reference . . . . .	4078
7.145src/main/activemq/transport/tcp/SslTransportFactory.h File Reference . .	4079
7.146src/main/activemq/transport/tcp/TcpTransport.h File Reference . . . . .	4079
7.147src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference .	4080
7.148src/main/activemq/transport/Transport.h File Reference . . . . .	4081
7.149src/main/activemq/transport/TransportFactory.h File Reference . . . . .	4081
7.150src/main/activemq/transport/TransportFilter.h File Reference . . . . .	4082
7.151src/main/activemq/transport/TransportListener.h File Reference . . . . .	4082
7.152src/main/activemq/transport/TransportRegistry.h File Reference . . . . .	4083
7.153src/main/activemq/util/ActiveMQProperties.h File Reference . . . . .	4083
7.154src/main/activemq/util/CMSExceptionSupport.h File Reference . . . . .	4084
7.154.1 Define Documentation . . . . .	4085
7.154.1.1 AMQ_CATCH_ALL_THROW_CMSEXCEPTION . . .	4085
7.155src/main/activemq/util/CompositeData.h File Reference . . . . .	4086
7.156src/main/activemq/util/Config.h File Reference . . . . .	4086
7.156.1 Define Documentation . . . . .	4086
7.156.1.1 AMQCPP_API . . . . .	4086
7.156.1.2 HAVE_PTHREAD_H . . . . .	4086
7.156.1.3 HAVE_UUID_T . . . . .	4086
7.156.1.4 HAVE_UUID_UUID_H . . . . .	4087
7.157src/main/cms/Config.h File Reference . . . . .	4087
7.157.1 Define Documentation . . . . .	4087
7.157.1.1 CMS_API . . . . .	4087
7.158src/main/decaf/util/Config.h File Reference . . . . .	4087
7.158.1 Define Documentation . . . . .	4087
7.158.1.1 DECAF_API . . . . .	4087
7.158.1.2 DECAF_UNUSED . . . . .	4087
7.158.1.3 HAVE_PTHREAD_H . . . . .	4087
7.158.1.4 HAVE_UUID_T . . . . .	4087
7.158.1.5 HAVE_UUID_UUID_H . . . . .	4087
7.159src/main/activemq/util/IdGenerator.h File Reference . . . . .	4087
7.160src/main/activemq/util/LongSequenceGenerator.h File Reference . . . .	4088
7.161src/main/activemq/util/MarshallingSupport.h File Reference . . . . .	4088



7.162src/main/activemq/util/MemoryUsage.h File Reference . . . . .	4089
7.163src/main/activemq/util/PrimitiveList.h File Reference . . . . .	4089
7.164src/main/activemq/util/PrimitiveMap.h File Reference . . . . .	4090
7.165src/main/activemq/util/PrimitiveValueConverter.h File Reference . . . . .	4090
7.166src/main/activemq/util/PrimitiveValueNode.h File Reference . . . . .	4091
7.167src/main/activemq/util/URISupport.h File Reference . . . . .	4091
7.168src/main/activemq/util/Usage.h File Reference . . . . .	4092
7.169src/main/activemq/wireformat/MarshalAware.h File Reference . . . . .	4092
7.170src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference . . . . .	4093
7.171src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference . . . . .	4093
7.172src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference . . . . .	4094
7.173src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4095
7.174src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4096
7.175src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4096
7.176src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4097
7.177src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4098
7.178src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarshaller.h File Reference . . . . .	4098
7.179src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4099
7.180src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4100
7.181src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4101
7.182src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4101
7.183src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4102
7.184src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h File Reference . . . . .	4103

7.185	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4104
7.186	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4104
7.187	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4105
7.188	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4106
7.189	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4106
7.190	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h	
	File Reference . . . . .	4107
7.191	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4108
7.192	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4109
7.193	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4109
7.194	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4110
7.195	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4111
7.196	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h	
	File Reference . . . . .	4112
7.197	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4112
7.198	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4113
7.199	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4114
7.200	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4114
7.201	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4115
7.202	src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h	
	File Reference . . . . .	4116
7.203	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4117
7.204	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h	
	File Reference . . . . .	4117

7.205src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h File Reference . . . . .	4118
7.206src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h File Reference . . . . .	4119
7.207src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h File Reference . . . . .	4120
7.208src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h File Reference . . . . .	4120
7.209src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h File Reference . . . . .	4121
7.210src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h File Reference . . . . .	4122
7.211src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h File Reference . . . . .	4122
7.212src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h File Reference . . . . .	4123
7.213src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h File Reference . . . . .	4124
7.214src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h File Reference . . . . .	4125
7.215src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h File Reference . . . . .	4125
7.216src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h File Reference . . . . .	4126
7.217src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h File Reference . . . . .	4127
7.218src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h File Reference . . . . .	4128
7.219src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h File Reference . . . . .	4128
7.220src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h File Reference . . . . .	4129
7.221src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h File Reference . . . . .	4130
7.222src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h File Reference . . . . .	4130
7.223src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference . . . . .	4131
7.224src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h File Reference . . . . .	4132

7.225src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4133
7.226src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempDestinationMarshaller.h	
File Reference . . . . .	4133
7.227src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4134
7.228src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4135
7.229src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4136
7.230src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4136
7.231src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4137
7.232src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempQueueMarshaller.h	
File Reference . . . . .	4138
7.233src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4139
7.234src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4139
7.235src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4140
7.236src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4141
7.237src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4141
7.238src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTempTopicMarshaller.h	
File Reference . . . . .	4142
7.239src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4143
7.240src/main/activemq/wireformat/openwire/marshall/v2/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4144
7.241src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4144
7.242src/main/activemq/wireformat/openwire/marshall/v4/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4145
7.243src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4146
7.244src/main/activemq/wireformat/openwire/marshall/v6/ActiveMQTextMessageMarshaller.h	
File Reference . . . . .	4147

7.245src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4147
7.246src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4148
7.247src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4149
7.248src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4149
7.249src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4150
7.250src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h	
File Reference . . . . .	4151
7.251src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h	
File Reference . . . . .	4152
7.252src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h	
File Reference . . . . .	4152
7.253src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h	
File Reference . . . . .	4153
7.254src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h	
File Reference . . . . .	4154
7.255src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h	
File Reference . . . . .	4155
7.256src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h	
File Reference . . . . .	4155
7.257src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h	
File Reference . . . . .	4156
7.258src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h	
File Reference . . . . .	4157
7.259src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h	
File Reference . . . . .	4157
7.260src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h	
File Reference . . . . .	4158
7.261src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h	
File Reference . . . . .	4159
7.262src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h	
File Reference . . . . .	4160
7.263src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h	
File Reference . . . . .	4160
7.264src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h	
File Reference . . . . .	4161

7.265src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h	
File Reference . . . . .	4162
7.266src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h	
File Reference . . . . .	4162
7.267src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h	
File Reference . . . . .	4163
7.268src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h	
File Reference . . . . .	4164
7.269src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h	
File Reference . . . . .	4164
7.270src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h	
File Reference . . . . .	4165
7.271src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h	
File Reference . . . . .	4166
7.272src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h	
File Reference . . . . .	4166
7.273src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h	
File Reference . . . . .	4167
7.274src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h	
File Reference . . . . .	4168
7.275src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h	
File Reference . . . . .	4169
7.276src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h	
File Reference . . . . .	4169
7.277src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h	
File Reference . . . . .	4170
7.278src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h	
File Reference . . . . .	4171
7.279src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h	
File Reference . . . . .	4171
7.280src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h	
File Reference . . . . .	4172
7.281src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h	
File Reference . . . . .	4173
7.282src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h	
File Reference . . . . .	4174
7.283src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h	
File Reference . . . . .	4174
7.284src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h	
File Reference . . . . .	4175

7.285src/main/activemq/wireformat/openwire/marshall/v5/ConnectionIdMarshaller.h	
File Reference . . . . .	4176
7.286src/main/activemq/wireformat/openwire/marshall/v6/ConnectionIdMarshaller.h	
File Reference . . . . .	4177
7.287src/main/activemq/wireformat/openwire/marshall/v1/ConnectionInfoMarshaller.h	
File Reference . . . . .	4177
7.288src/main/activemq/wireformat/openwire/marshall/v2/ConnectionInfoMarshaller.h	
File Reference . . . . .	4178
7.289src/main/activemq/wireformat/openwire/marshall/v3/ConnectionInfoMarshaller.h	
File Reference . . . . .	4179
7.290src/main/activemq/wireformat/openwire/marshall/v4/ConnectionInfoMarshaller.h	
File Reference . . . . .	4179
7.291src/main/activemq/wireformat/openwire/marshall/v5/ConnectionInfoMarshaller.h	
File Reference . . . . .	4180
7.292src/main/activemq/wireformat/openwire/marshall/v6/ConnectionInfoMarshaller.h	
File Reference . . . . .	4181
7.293src/main/activemq/wireformat/openwire/marshall/v1/ConsumerControlMarshaller.h	
File Reference . . . . .	4182
7.294src/main/activemq/wireformat/openwire/marshall/v2/ConsumerControlMarshaller.h	
File Reference . . . . .	4182
7.295src/main/activemq/wireformat/openwire/marshall/v3/ConsumerControlMarshaller.h	
File Reference . . . . .	4183
7.296src/main/activemq/wireformat/openwire/marshall/v4/ConsumerControlMarshaller.h	
File Reference . . . . .	4184
7.297src/main/activemq/wireformat/openwire/marshall/v5/ConsumerControlMarshaller.h	
File Reference . . . . .	4185
7.298src/main/activemq/wireformat/openwire/marshall/v6/ConsumerControlMarshaller.h	
File Reference . . . . .	4185
7.299src/main/activemq/wireformat/openwire/marshall/v1/ConsumerIdMarshaller.h	
File Reference . . . . .	4186
7.300src/main/activemq/wireformat/openwire/marshall/v2/ConsumerIdMarshaller.h	
File Reference . . . . .	4187
7.301src/main/activemq/wireformat/openwire/marshall/v3/ConsumerIdMarshaller.h	
File Reference . . . . .	4187
7.302src/main/activemq/wireformat/openwire/marshall/v4/ConsumerIdMarshaller.h	
File Reference . . . . .	4188
7.303src/main/activemq/wireformat/openwire/marshall/v5/ConsumerIdMarshaller.h	
File Reference . . . . .	4189
7.304src/main/activemq/wireformat/openwire/marshall/v6/ConsumerIdMarshaller.h	
File Reference . . . . .	4190

7.305src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h	
File Reference . . . . .	4190
7.306src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h	
File Reference . . . . .	4191
7.307src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	
File Reference . . . . .	4192
7.308src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	
File Reference . . . . .	4193
7.309src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h	
File Reference . . . . .	4193
7.310src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h	
File Reference . . . . .	4194
7.311src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
File Reference . . . . .	4195
7.312src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h	
File Reference . . . . .	4195
7.313src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
File Reference . . . . .	4196
7.314src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
File Reference . . . . .	4197
7.315src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
File Reference . . . . .	4198
7.316src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h	
File Reference . . . . .	4198
7.317src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
File Reference . . . . .	4199
7.318src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h	
File Reference . . . . .	4200
7.319src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h	
File Reference . . . . .	4201
7.320src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h	
File Reference . . . . .	4201
7.321src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h	
File Reference . . . . .	4202
7.322src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h	
File Reference . . . . .	4203
7.323src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h	
File Reference . . . . .	4203
7.324src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h	
File Reference . . . . .	4204



7.325src/main/activemq/wireformat/openwire/marshall/v3/DataResponseMarshaller.h File Reference . . . . .	4205
7.326src/main/activemq/wireformat/openwire/marshall/v4/DataResponseMarshaller.h File Reference . . . . .	4206
7.327src/main/activemq/wireformat/openwire/marshall/v5/DataResponseMarshaller.h File Reference . . . . .	4206
7.328src/main/activemq/wireformat/openwire/marshall/v6/DataResponseMarshaller.h File Reference . . . . .	4207
7.329src/main/activemq/wireformat/openwire/marshall/v1/DestinationInfoMarshaller.h File Reference . . . . .	4208
7.330src/main/activemq/wireformat/openwire/marshall/v2/DestinationInfoMarshaller.h File Reference . . . . .	4209
7.331src/main/activemq/wireformat/openwire/marshall/v3/DestinationInfoMarshaller.h File Reference . . . . .	4209
7.332src/main/activemq/wireformat/openwire/marshall/v4/DestinationInfoMarshaller.h File Reference . . . . .	4210
7.333src/main/activemq/wireformat/openwire/marshall/v5/DestinationInfoMarshaller.h File Reference . . . . .	4211
7.334src/main/activemq/wireformat/openwire/marshall/v6/DestinationInfoMarshaller.h File Reference . . . . .	4211
7.335src/main/activemq/wireformat/openwire/marshall/v1/DiscoveryEventMarshaller.h File Reference . . . . .	4212
7.336src/main/activemq/wireformat/openwire/marshall/v2/DiscoveryEventMarshaller.h File Reference . . . . .	4213
7.337src/main/activemq/wireformat/openwire/marshall/v3/DiscoveryEventMarshaller.h File Reference . . . . .	4214
7.338src/main/activemq/wireformat/openwire/marshall/v4/DiscoveryEventMarshaller.h File Reference . . . . .	4214
7.339src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEventMarshaller.h File Reference . . . . .	4215
7.340src/main/activemq/wireformat/openwire/marshall/v6/DiscoveryEventMarshaller.h File Reference . . . . .	4216
7.341src/main/activemq/wireformat/openwire/marshall/v1/ExceptionResponseMarshaller.h File Reference . . . . .	4217
7.342src/main/activemq/wireformat/openwire/marshall/v2/ExceptionResponseMarshaller.h File Reference . . . . .	4217
7.343src/main/activemq/wireformat/openwire/marshall/v3/ExceptionResponseMarshaller.h File Reference . . . . .	4218
7.344src/main/activemq/wireformat/openwire/marshall/v4/ExceptionResponseMarshaller.h File Reference . . . . .	4219

7.345src/main/activemq/wireformat/openwire/marshall/v5/ExceptionResponseMarshaller.h	
File Reference . . . . .	4219
7.346src/main/activemq/wireformat/openwire/marshall/v6/ExceptionResponseMarshaller.h	
File Reference . . . . .	4220
7.347src/main/activemq/wireformat/openwire/marshall/v1/FlushCommandMarshaller.h	
File Reference . . . . .	4221
7.348src/main/activemq/wireformat/openwire/marshall/v2/FlushCommandMarshaller.h	
File Reference . . . . .	4222
7.349src/main/activemq/wireformat/openwire/marshall/v3/FlushCommandMarshaller.h	
File Reference . . . . .	4222
7.350src/main/activemq/wireformat/openwire/marshall/v4/FlushCommandMarshaller.h	
File Reference . . . . .	4223
7.351src/main/activemq/wireformat/openwire/marshall/v5/FlushCommandMarshaller.h	
File Reference . . . . .	4224
7.352src/main/activemq/wireformat/openwire/marshall/v6/FlushCommandMarshaller.h	
File Reference . . . . .	4225
7.353src/main/activemq/wireformat/openwire/marshall/v1/IntegerResponseMarshaller.h	
File Reference . . . . .	4225
7.354src/main/activemq/wireformat/openwire/marshall/v2/IntegerResponseMarshaller.h	
File Reference . . . . .	4226
7.355src/main/activemq/wireformat/openwire/marshall/v3/IntegerResponseMarshaller.h	
File Reference . . . . .	4227
7.356src/main/activemq/wireformat/openwire/marshall/v4/IntegerResponseMarshaller.h	
File Reference . . . . .	4227
7.357src/main/activemq/wireformat/openwire/marshall/v5/IntegerResponseMarshaller.h	
File Reference . . . . .	4228
7.358src/main/activemq/wireformat/openwire/marshall/v6/IntegerResponseMarshaller.h	
File Reference . . . . .	4229
7.359src/main/activemq/wireformat/openwire/marshall/v1/JournalQueueAckMarshaller.h	
File Reference . . . . .	4230
7.360src/main/activemq/wireformat/openwire/marshall/v2/JournalQueueAckMarshaller.h	
File Reference . . . . .	4230
7.361src/main/activemq/wireformat/openwire/marshall/v3/JournalQueueAckMarshaller.h	
File Reference . . . . .	4231
7.362src/main/activemq/wireformat/openwire/marshall/v4/JournalQueueAckMarshaller.h	
File Reference . . . . .	4232
7.363src/main/activemq/wireformat/openwire/marshall/v5/JournalQueueAckMarshaller.h	
File Reference . . . . .	4233
7.364src/main/activemq/wireformat/openwire/marshall/v6/JournalQueueAckMarshaller.h	
File Reference . . . . .	4233

7.365src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
File Reference . . . . .	4234
7.366src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h	
File Reference . . . . .	4235
7.367src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
File Reference . . . . .	4235
7.368src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
File Reference . . . . .	4236
7.369src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
File Reference . . . . .	4237
7.370src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h	
File Reference . . . . .	4238
7.371src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h	
File Reference . . . . .	4238
7.372src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h	
File Reference . . . . .	4239
7.373src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	
File Reference . . . . .	4240
7.374src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	
File Reference . . . . .	4241
7.375src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h	
File Reference . . . . .	4241
7.376src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h	
File Reference . . . . .	4242
7.377src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
File Reference . . . . .	4243
7.378src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h	
File Reference . . . . .	4243
7.379src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	
File Reference . . . . .	4244
7.380src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
File Reference . . . . .	4245
7.381src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
File Reference . . . . .	4246
7.382src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller.h	
File Reference . . . . .	4246
7.383src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4247
7.384src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h	
File Reference . . . . .	4248

7.385src/main/activemq/wireformat/openwire/marshall/v3/KeepAliveInfoMarshaller.h File Reference . . . . .	4249
7.386src/main/activemq/wireformat/openwire/marshall/v4/KeepAliveInfoMarshaller.h File Reference . . . . .	4249
7.387src/main/activemq/wireformat/openwire/marshall/v5/KeepAliveInfoMarshaller.h File Reference . . . . .	4250
7.388src/main/activemq/wireformat/openwire/marshall/v6/KeepAliveInfoMarshaller.h File Reference . . . . .	4251
7.389src/main/activemq/wireformat/openwire/marshall/v1/LastPartialCommandMarshaller.h File Reference . . . . .	4251
7.390src/main/activemq/wireformat/openwire/marshall/v2/LastPartialCommandMarshaller.h File Reference . . . . .	4252
7.391src/main/activemq/wireformat/openwire/marshall/v3/LastPartialCommandMarshaller.h File Reference . . . . .	4253
7.392src/main/activemq/wireformat/openwire/marshall/v4/LastPartialCommandMarshaller.h File Reference . . . . .	4254
7.393src/main/activemq/wireformat/openwire/marshall/v5/LastPartialCommandMarshaller.h File Reference . . . . .	4254
7.394src/main/activemq/wireformat/openwire/marshall/v6/LastPartialCommandMarshaller.h File Reference . . . . .	4255
7.395src/main/activemq/wireformat/openwire/marshall/v1/LocalTransactionIdMarshaller.h File Reference . . . . .	4256
7.396src/main/activemq/wireformat/openwire/marshall/v2/LocalTransactionIdMarshaller.h File Reference . . . . .	4257
7.397src/main/activemq/wireformat/openwire/marshall/v3/LocalTransactionIdMarshaller.h File Reference . . . . .	4257
7.398src/main/activemq/wireformat/openwire/marshall/v4/LocalTransactionIdMarshaller.h File Reference . . . . .	4258
7.399src/main/activemq/wireformat/openwire/marshall/v5/LocalTransactionIdMarshaller.h File Reference . . . . .	4259
7.400src/main/activemq/wireformat/openwire/marshall/v6/LocalTransactionIdMarshaller.h File Reference . . . . .	4259
7.401src/main/activemq/wireformat/openwire/marshall/v1/MarshallerFactory.h File Reference . . . . .	4260
7.402src/main/activemq/wireformat/openwire/marshall/v2/MarshallerFactory.h File Reference . . . . .	4261
7.403src/main/activemq/wireformat/openwire/marshall/v3/MarshallerFactory.h File Reference . . . . .	4261
7.404src/main/activemq/wireformat/openwire/marshall/v4/MarshallerFactory.h File Reference . . . . .	4262

7.405src/main/activemq/wireformat/openwire/marshall/v5/MarshallFactory.h File Reference . . . . .	4262
7.406src/main/activemq/wireformat/openwire/marshall/v6/MarshallFactory.h File Reference . . . . .	4263
7.407src/main/activemq/wireformat/openwire/marshall/v1/MessageAckMarshaller.h File Reference . . . . .	4263
7.408src/main/activemq/wireformat/openwire/marshall/v2/MessageAckMarshaller.h File Reference . . . . .	4264
7.409src/main/activemq/wireformat/openwire/marshall/v3/MessageAckMarshaller.h File Reference . . . . .	4265
7.410src/main/activemq/wireformat/openwire/marshall/v4/MessageAckMarshaller.h File Reference . . . . .	4265
7.411src/main/activemq/wireformat/openwire/marshall/v5/MessageAckMarshaller.h File Reference . . . . .	4266
7.412src/main/activemq/wireformat/openwire/marshall/v6/MessageAckMarshaller.h File Reference . . . . .	4267
7.413src/main/activemq/wireformat/openwire/marshall/v1/MessageDispatchMarshaller.h File Reference . . . . .	4267
7.414src/main/activemq/wireformat/openwire/marshall/v2/MessageDispatchMarshaller.h File Reference . . . . .	4268
7.415src/main/activemq/wireformat/openwire/marshall/v3/MessageDispatchMarshaller.h File Reference . . . . .	4269
7.416src/main/activemq/wireformat/openwire/marshall/v4/MessageDispatchMarshaller.h File Reference . . . . .	4270
7.417src/main/activemq/wireformat/openwire/marshall/v5/MessageDispatchMarshaller.h File Reference . . . . .	4270
7.418src/main/activemq/wireformat/openwire/marshall/v6/MessageDispatchMarshaller.h File Reference . . . . .	4271
7.419src/main/activemq/wireformat/openwire/marshall/v1/MessageDispatchNotificationMarshaller.h File Reference . . . . .	4272
7.420src/main/activemq/wireformat/openwire/marshall/v2/MessageDispatchNotificationMarshaller.h File Reference . . . . .	4273
7.421src/main/activemq/wireformat/openwire/marshall/v3/MessageDispatchNotificationMarshaller.h File Reference . . . . .	4273
7.422src/main/activemq/wireformat/openwire/marshall/v4/MessageDispatchNotificationMarshaller.h File Reference . . . . .	4274
7.423src/main/activemq/wireformat/openwire/marshall/v5/MessageDispatchNotificationMarshaller.h File Reference . . . . .	4275
7.424src/main/activemq/wireformat/openwire/marshall/v6/MessageDispatchNotificationMarshaller.h File Reference . . . . .	4276

7.425src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h	
File Reference . . . . .	4276
7.426src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h	
File Reference . . . . .	4277
7.427src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h	
File Reference . . . . .	4278
7.428src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h	
File Reference . . . . .	4278
7.429src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h	
File Reference . . . . .	4279
7.430src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h	
File Reference . . . . .	4280
7.431src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h	
File Reference . . . . .	4281
7.432src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h	
File Reference . . . . .	4281
7.433src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h	
File Reference . . . . .	4282
7.434src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h	
File Reference . . . . .	4283
7.435src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h	
File Reference . . . . .	4283
7.436src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h	
File Reference . . . . .	4284
7.437src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h	
File Reference . . . . .	4285
7.438src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h	
File Reference . . . . .	4285
7.439src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h	
File Reference . . . . .	4286
7.440src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h	
File Reference . . . . .	4287
7.441src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h	
File Reference . . . . .	4287
7.442src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h	
File Reference . . . . .	4288
7.443src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4289
7.444src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4290

7.445src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4290
7.446src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4291
7.447src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4292
7.448src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h	
File Reference . . . . .	4293
7.449src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h	
File Reference . . . . .	4293
7.450src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h	
File Reference . . . . .	4294
7.451src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h	
File Reference . . . . .	4295
7.452src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h	
File Reference . . . . .	4295
7.453src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h	
File Reference . . . . .	4296
7.454src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h	
File Reference . . . . .	4297
7.455src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h	
File Reference . . . . .	4298
7.456src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h	
File Reference . . . . .	4298
7.457src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h	
File Reference . . . . .	4299
7.458src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h	
File Reference . . . . .	4300
7.459src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h	
File Reference . . . . .	4301
7.460src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h	
File Reference . . . . .	4301
7.461src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h	
File Reference . . . . .	4302
7.462src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h	
File Reference . . . . .	4303
7.463src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h	
File Reference . . . . .	4303
7.464src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h	
File Reference . . . . .	4304

7.465src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h	
File Reference . . . . .	4305
7.466src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h	
File Reference . . . . .	4306
7.467src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h	
File Reference . . . . .	4306
7.468src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h	
File Reference . . . . .	4307
7.469src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h	
File Reference . . . . .	4308
7.470src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h	
File Reference . . . . .	4308
7.471src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h	
File Reference . . . . .	4309
7.472src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h	
File Reference . . . . .	4310
7.473src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h	
File Reference . . . . .	4311
7.474src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h	
File Reference . . . . .	4311
7.475src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h	
File Reference . . . . .	4312
7.476src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h	
File Reference . . . . .	4313
7.477src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h	
File Reference . . . . .	4313
7.478src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h	
File Reference . . . . .	4314
7.479src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4315
7.480src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4315
7.481src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4316
7.482src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4317
7.483src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4317
7.484src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h	
File Reference . . . . .	4318



7.485src/main/activemq/wireformat/openwire/marshall/v1/ReplayCommandMarshaller.h	
File Reference . . . . .	4319
7.486src/main/activemq/wireformat/openwire/marshall/v2/ReplayCommandMarshaller.h	
File Reference . . . . .	4320
7.487src/main/activemq/wireformat/openwire/marshall/v3/ReplayCommandMarshaller.h	
File Reference . . . . .	4320
7.488src/main/activemq/wireformat/openwire/marshall/v4/ReplayCommandMarshaller.h	
File Reference . . . . .	4321
7.489src/main/activemq/wireformat/openwire/marshall/v5/ReplayCommandMarshaller.h	
File Reference . . . . .	4322
7.490src/main/activemq/wireformat/openwire/marshall/v6/ReplayCommandMarshaller.h	
File Reference . . . . .	4323
7.491src/main/activemq/wireformat/openwire/marshall/v1/ResponseMarshaller.h	
File Reference . . . . .	4323
7.492src/main/activemq/wireformat/openwire/marshall/v2/ResponseMarshaller.h	
File Reference . . . . .	4324
7.493src/main/activemq/wireformat/openwire/marshall/v3/ResponseMarshaller.h	
File Reference . . . . .	4325
7.494src/main/activemq/wireformat/openwire/marshall/v4/ResponseMarshaller.h	
File Reference . . . . .	4325
7.495src/main/activemq/wireformat/openwire/marshall/v5/ResponseMarshaller.h	
File Reference . . . . .	4326
7.496src/main/activemq/wireformat/openwire/marshall/v6/ResponseMarshaller.h	
File Reference . . . . .	4327
7.497src/main/activemq/wireformat/openwire/marshall/v1/SessionIdMarshaller.h	
File Reference . . . . .	4328
7.498src/main/activemq/wireformat/openwire/marshall/v2/SessionIdMarshaller.h	
File Reference . . . . .	4328
7.499src/main/activemq/wireformat/openwire/marshall/v3/SessionIdMarshaller.h	
File Reference . . . . .	4329
7.500src/main/activemq/wireformat/openwire/marshall/v4/SessionIdMarshaller.h	
File Reference . . . . .	4330
7.501src/main/activemq/wireformat/openwire/marshall/v5/SessionIdMarshaller.h	
File Reference . . . . .	4330
7.502src/main/activemq/wireformat/openwire/marshall/v6/SessionIdMarshaller.h	
File Reference . . . . .	4331
7.503src/main/activemq/wireformat/openwire/marshall/v1/SessionInfoMarshaller.h	
File Reference . . . . .	4332
7.504src/main/activemq/wireformat/openwire/marshall/v2/SessionInfoMarshaller.h	
File Reference . . . . .	4332

7.505src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h	
File Reference . . . . .	4333
7.506src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h	
File Reference . . . . .	4334
7.507src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h	
File Reference . . . . .	4334
7.508src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h	
File Reference . . . . .	4335
7.509src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h	
File Reference . . . . .	4336
7.510src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h	
File Reference . . . . .	4336
7.511src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h	
File Reference . . . . .	4337
7.512src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h	
File Reference . . . . .	4338
7.513src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h	
File Reference . . . . .	4338
7.514src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h	
File Reference . . . . .	4339
7.515src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4340
7.516src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4341
7.517src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4341
7.518src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4342
7.519src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4343
7.520src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h	
File Reference . . . . .	4344
7.521src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h	
File Reference . . . . .	4344
7.522src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h	
File Reference . . . . .	4345
7.523src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h	
File Reference . . . . .	4346
7.524src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h	
File Reference . . . . .	4346

7.525src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h	
File Reference . . . . .	4347
7.526src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h	
File Reference . . . . .	4348
7.527src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h	
File Reference . . . . .	4349
7.528src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h	
File Reference . . . . .	4349
7.529src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h	
File Reference . . . . .	4350
7.530src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h	
File Reference . . . . .	4351
7.531src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h	
File Reference . . . . .	4352
7.532src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h	
File Reference . . . . .	4352
7.533src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h	
File Reference . . . . .	4353
7.534src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h	
File Reference . . . . .	4354
7.535src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h	
File Reference . . . . .	4354
7.536src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h	
File Reference . . . . .	4355
7.537src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h	
File Reference . . . . .	4356
7.538src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h	
File Reference . . . . .	4357
7.539src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h	
File Reference . . . . .	4357
7.540src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h	
File Reference . . . . .	4358
7.541src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h	
File Reference . . . . .	4359
7.542src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h	
File Reference . . . . .	4360
7.543src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h	
File Reference . . . . .	4360
7.544src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h	
File Reference . . . . .	4361

7.545src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference	4362
7.546src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference . . . . .	4362
7.547src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference . . . . .	4363
7.548src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference . . . . .	4364
7.549src/main/activemq/wireformat/openwire/Utils/BooleanStream.h File Reference . . . . .	4364
7.550src/main/activemq/wireformat/openwire/Utils/HexTable.h File Reference . . . . .	4365
7.551src/main/activemq/wireformat/openwire/Utils/MessagePropertyInterceptor.h File Reference . . . . .	4365
7.552src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference . . . . .	4366
7.553src/main/activemq/wireformat/stomp/StompFrame.h File Reference . . . . .	4366
7.554src/main/activemq/wireformat/stomp/StompHelper.h File Reference . . . . .	4367
7.555src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference . . . . .	4368
7.556src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference . . . . .	4368
7.557src/main/activemq/wireformat/WireFormat.h File Reference . . . . .	4369
7.558src/main/activemq/wireformat/WireFormatFactory.h File Reference . . . . .	4370
7.559src/main/activemq/wireformat/WireFormatNegotiator.h File Reference . . . . .	4370
7.560src/main/activemq/wireformat/WireFormatRegistry.h File Reference . . . . .	4371
7.561src/main/cms/BytesMessage.h File Reference . . . . .	4371
7.562src/main/cms/Closeable.h File Reference . . . . .	4372
7.563src/main/decaf/io/Closeable.h File Reference . . . . .	4372
7.564src/main/cms/CMSException.h File Reference . . . . .	4373
7.565src/main/cms/CMSProperties.h File Reference . . . . .	4373
7.566src/main/cms/CMSSecurityException.h File Reference . . . . .	4374
7.567src/main/cms/Connection.h File Reference . . . . .	4374
7.568src/main/cms/ConnectionFactory.h File Reference . . . . .	4375
7.569src/main/cms/ConnectionMetaData.h File Reference . . . . .	4375
7.570src/main/cms/DeliveryMode.h File Reference . . . . .	4375
7.571src/main/cms/Destination.h File Reference . . . . .	4376
7.572src/main/cms/ExceptionListener.h File Reference . . . . .	4376
7.573src/main/cms/IllegalStateException.h File Reference . . . . .	4377

7.574src/main/decaf/lang/exceptions/IllegalStateException.h File Reference . . . . .	4377
7.575src/main/cms/InvalidClientIdException.h File Reference . . . . .	4378
7.576src/main/cms/InvalidDestinationException.h File Reference . . . . .	4378
7.577src/main/cms/InvalidSelectorException.h File Reference . . . . .	4378
7.578src/main/cms/MapMessage.h File Reference . . . . .	4379
7.579src/main/cms/MessageConsumer.h File Reference . . . . .	4379
7.580src/main/cms/MessageEnumeration.h File Reference . . . . .	4380
7.581src/main/cms/MessageEOFException.h File Reference . . . . .	4380
7.582src/main/cms/MessageFormatException.h File Reference . . . . .	4381
7.583src/main/cms/MessageListener.h File Reference . . . . .	4381
7.584src/main/cms/MessageNotReadableException.h File Reference . . . . .	4382
7.585src/main/cms/MessageNotWriteableException.h File Reference . . . . .	4382
7.586src/main/cms/MessageProducer.h File Reference . . . . .	4382
7.587src/main/cms/ObjectMessage.h File Reference . . . . .	4383
7.588src/main/cms/Queue.h File Reference . . . . .	4383
7.589src/main/decaf/util/Queue.h File Reference . . . . .	4384
7.590src/main/cms/QueueBrowser.h File Reference . . . . .	4384
7.591src/main/cms/Session.h File Reference . . . . .	4385
7.592src/main/cms/Startable.h File Reference . . . . .	4386
7.593src/main/cms/Stopable.h File Reference . . . . .	4386
7.594src/main/cms/StreamMessage.h File Reference . . . . .	4387
7.595src/main/cms/TemporaryQueue.h File Reference . . . . .	4387
7.596src/main/cms/TemporaryTopic.h File Reference . . . . .	4388
7.597src/main/cms/TextMessage.h File Reference . . . . .	4388
7.598src/main/cms/Topic.h File Reference . . . . .	4388
7.599src/main/cms/UnsupportedOperationException.h File Reference . . . . .	4389
7.600src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference . . . . .	4389
7.601src/main/decaf/internal/AprPool.h File Reference . . . . .	4390
7.602src/main/decaf/internal/DecafRuntime.h File Reference . . . . .	4390
7.603src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference . . . . .	4391
7.604src/main/decaf/internal/io/StandardInputStream.h File Reference . . . . .	4391
7.605src/main/decaf/internal/io/StandardOutputStream.h File Reference . . . . .	4392
7.606src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference . . . . .	4392

7.607src/main/decaf/internal/net/DefaultSocketFactory.h File Reference . . . .	4393
7.608src/main/decaf/internal/net/Network.h File Reference . . . . .	4393
7.609src/main/decaf/internal/net/SocketFileDescriptor.h File Reference . . . .	4394
7.610src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference . . .	4394
7.611src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference . . . . .	4395
7.612src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference	4395
7.613src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference . . . . .	4396
7.614src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference . . . . .	4396
7.615src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference . . . . .	4397
7.616src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference . . . . .	4397
7.617src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference	4398
7.618src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference . . . . .	4398
7.619src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference . . . . .	4399
7.620src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference . . . . .	4399
7.621src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference . . . . .	4400
7.622src/main/decaf/internal/net/tcp/TcpSocket.h File Reference . . . . .	4400
7.623src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference .	4401
7.624src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference	4402
7.625src/main/decaf/internal/net/URIEncoderDecoder.h File Reference . . . .	4402
7.626src/main/decaf/internal/net/URIHelper.h File Reference . . . . .	4403
7.627src/main/decaf/internal/net/URIType.h File Reference . . . . .	4403
7.628src/main/decaf/internal/nio/BufferFactory.h File Reference . . . . .	4404
7.629src/main/decaf/internal/nio/ByteBuffer.h File Reference . . . . .	4404
7.630src/main/decaf/internal/nio/CharArrayBuffer.h File Reference . . . . .	4405
7.631src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference . . . .	4406
7.632src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference . . . . .	4406
7.633src/main/decaf/internal/nio/IntArrayBuffer.h File Reference . . . . .	4407
7.634src/main/decaf/internal/nio/LongArrayBuffer.h File Reference . . . . .	4408

7.635src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference . . . . .	4408
7.636src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference	4409
7.637src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference . . . . .	4409
7.638src/main/decaf/internal/util/ByteArrayAdapter.h File Reference . . . . .	4410
7.639src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference . . . . .	4411
7.640src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference . . . . .	4411
7.641src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference . . . . .	4412
7.642src/main/decaf/internal/util/concurrent/Transferer.h File Reference . . . . .	4412
7.643src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference . . . . .	4413
7.644src/main/decaf/internal/util/concurrent/TransferStack.h File Reference . . . . .	4413
7.645src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference . . . . .	4414
7.646src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference . . . . .	4414
7.647src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference	4415
7.648src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference . . . . .	4415
7.649src/main/decaf/internal/util/GenericResource.h File Reference . . . . .	4415
7.650src/main/decaf/internal/util/HexStringParser.h File Reference . . . . .	4416
7.651src/main/decaf/internal/util/Resource.h File Reference . . . . .	4416
7.652src/main/decaf/internal/util/TimerTaskHeap.h File Reference . . . . .	4417
7.653src/main/decaf/internal/util/zip/crc32.h File Reference . . . . .	4417
7.653.1 Variable Documentation . . . . .	4417
7.653.1.1 crc_table . . . . .	4417
7.654src/main/decaf/internal/util/zip/deflate.h File Reference . . . . .	4417
7.654.1 Define Documentation . . . . .	4419
7.654.1.1 _tr_tally_dist . . . . .	4419
7.654.1.2 _tr_tally_lit . . . . .	4419
7.654.1.3 BL_CODES . . . . .	4419
7.654.1.4 BUSY_STATE . . . . .	4419
7.654.1.5 Code . . . . .	4419
7.654.1.6 COMMENT_STATE . . . . .	4419
7.654.1.7 d_code . . . . .	4420

7.654.1.8 D_CODES . . . . .	4420
7.654.1.9 Dad . . . . .	4420
7.654.1.10EXTRA_STATE . . . . .	4420
7.654.1.11FINISH_STATE . . . . .	4420
7.654.1.12Freq . . . . .	4420
7.654.1.13GZIP . . . . .	4420
7.654.1.14HCRC_STATE . . . . .	4420
7.654.1.15HEAP_SIZE . . . . .	4420
7.654.1.16NIT_STATE . . . . .	4420
7.654.1.17L_CODES . . . . .	4420
7.654.1.18Len . . . . .	4420
7.654.1.19LENGTH_CODES . . . . .	4420
7.654.1.20LITERALS . . . . .	4420
7.654.1.21MAX_BITS . . . . .	4420
7.654.1.22MAX_DIST . . . . .	4420
7.654.1.23max_insert_length . . . . .	4420
7.654.1.24MIN_LOOKAHEAD . . . . .	4420
7.654.1.25NAME_STATE . . . . .	4420
7.654.1.26put_byte . . . . .	4420
7.654.1.27WIN_INIT . . . . .	4420
7.654.2 Typedef Documentation . . . . .	4420
7.654.2.1 ct_data . . . . .	4420
7.654.2.2 deflate_state . . . . .	4421
7.654.2.3 IPos . . . . .	4421
7.654.2.4 Pos . . . . .	4421
7.654.2.5 Posf . . . . .	4421
7.654.2.6 static_tree_desc . . . . .	4421
7.654.2.7 tree_desc . . . . .	4421
7.654.3 Function Documentation . . . . .	4421
7.654.3.1 OF . . . . .	4421
7.654.3.2 OF . . . . .	4421
7.654.3.3 OF . . . . .	4421
7.654.4 Variable Documentation . . . . .	4421
7.654.4.1 _dist_code . . . . .	4421



7.654.4.2 _length_code . . . . .	4421
7.655src/main/decaf/internal/util/zip/gzguts.h File Reference . . . . .	4421
7.655.1 Define Documentation . . . . .	4422
7.655.1.1 COPY . . . . .	4422
7.655.1.2 GT_OFF . . . . .	4422
7.655.1.3 GZ_APPEND . . . . .	4422
7.655.1.4 GZ_NONE . . . . .	4422
7.655.1.5 GZ_READ . . . . .	4422
7.655.1.6 GZ_WRITE . . . . .	4422
7.655.1.7 GZBUFSIZE . . . . .	4422
7.655.1.8 GZIP . . . . .	4422
7.655.1.9 local . . . . .	4422
7.655.1.10LOOK . . . . .	4422
7.655.1.11ZLIB_INTERNAL . . . . .	4422
7.655.1.12zstrerror . . . . .	4423
7.655.2 Typedef Documentation . . . . .	4423
7.655.2.1 gz_statep . . . . .	4423
7.655.3 Function Documentation . . . . .	4423
7.655.3.1 OF . . . . .	4423
7.655.3.2 OF . . . . .	4423
7.655.3.3 OF . . . . .	4423
7.655.3.4 OF . . . . .	4423
7.655.3.5 OF . . . . .	4423
7.655.3.6 OF . . . . .	4423
7.655.3.7 OF . . . . .	4423
7.656src/main/decaf/internal/util/zip/inffast.h File Reference . . . . .	4423
7.656.1 Function Documentation . . . . .	4423
7.656.1.1 OF . . . . .	4423
7.657src/main/decaf/internal/util/zip/inffixed.h File Reference . . . . .	4423
7.658src/main/decaf/internal/util/zip/inflate.h File Reference . . . . .	4423
7.658.1 Define Documentation . . . . .	4424
7.658.1.1 GUNZIP . . . . .	4424
7.658.2 Enumeration Type Documentation . . . . .	4424
7.658.2.1 inflate_mode . . . . .	4424

7.659src/main/decaf/internal/util/zip/infrees.h File Reference . . . . .	4425
7.659.1 Define Documentation . . . . .	4426
7.659.1.1 ENOUGH . . . . .	4426
7.659.1.2 ENOUGH_DISTS . . . . .	4426
7.659.1.3 ENOUGH_LENS . . . . .	4426
7.659.2 Enumeration Type Documentation . . . . .	4426
7.659.2.1 codetype . . . . .	4426
7.659.3 Function Documentation . . . . .	4426
7.659.3.1 OF . . . . .	4426
7.660src/main/decaf/internal/util/zip/trees.h File Reference . . . . .	4426
7.660.1 Variable Documentation . . . . .	4426
7.660.1.1 _dist_code . . . . .	4426
7.660.1.2 _length_code . . . . .	4427
7.660.1.3 base_dist . . . . .	4427
7.660.1.4 base_length . . . . .	4427
7.660.1.5 static_dtree . . . . .	4428
7.660.1.6 static_ltree . . . . .	4428
7.661src/main/decaf/internal/util/zip/zconf.h File Reference . . . . .	4428
7.661.1 Define Documentation . . . . .	4429
7.661.1.1 const . . . . .	4429
7.661.1.2 FAR . . . . .	4429
7.661.1.3 MAX_MEM_LEVEL . . . . .	4429
7.661.1.4 MAX_WBITS . . . . .	4429
7.661.1.5 OF . . . . .	4429
7.661.1.6 SEEK_CUR . . . . .	4429
7.661.1.7 SEEK_END . . . . .	4429
7.661.1.8 SEEK_SET . . . . .	4429
7.661.1.9 z_off64_t . . . . .	4429
7.661.1.10z_off_t . . . . .	4429
7.661.1.11ZEXPORT . . . . .	4429
7.661.1.12ZEXPORTVA . . . . .	4429
7.661.1.13ZEXTERN . . . . .	4429
7.661.2 Typedef Documentation . . . . .	4429
7.661.2.1 Byte . . . . .	4429

7.661.2.2 Bytef . . . . .	4429
7.661.2.3 charf . . . . .	4429
7.661.2.4 intf . . . . .	4429
7.661.2.5 ulnt . . . . .	4430
7.661.2.6 ulntf . . . . .	4430
7.661.2.7 uLong . . . . .	4430
7.661.2.8 uLongf . . . . .	4430
7.661.2.9 voidp . . . . .	4430
7.661.2.10voidpc . . . . .	4430
7.661.2.11voidpf . . . . .	4430
7.662src/main/decaf/internal/util/zip/zlib.h File Reference . . . . .	4430
7.662.1 Define Documentation . . . . .	4433
7.662.1.1 deflateInit . . . . .	4433
7.662.1.2 deflateInit2 . . . . .	4433
7.662.1.3 inflateBackInit . . . . .	4433
7.662.1.4 inflateInit . . . . .	4433
7.662.1.5 inflateInit2 . . . . .	4433
7.662.1.6 Z_ASCII . . . . .	4433
7.662.1.7 Z_BEST_COMPRESSION . . . . .	4433
7.662.1.8 Z_BEST_SPEED . . . . .	4433
7.662.1.9 Z_BINARY . . . . .	4433
7.662.1.10Z_BLOCK . . . . .	4433
7.662.1.11Z_BUF_ERROR . . . . .	4433
7.662.1.12Z_DATA_ERROR . . . . .	4433
7.662.1.13Z_DEFAULT_COMPRESSION . . . . .	4433
7.662.1.14Z_DEFAULT_STRATEGY . . . . .	4433
7.662.1.15Z_DEFLATED . . . . .	4433
7.662.1.16Z_ERRNO . . . . .	4433
7.662.1.17Z_FILTERED . . . . .	4433
7.662.1.18Z_FINISH . . . . .	4434
7.662.1.19Z_FIXED . . . . .	4434
7.662.1.20Z_FULL_FLUSH . . . . .	4434
7.662.1.21Z_HUFFMAN_ONLY . . . . .	4434
7.662.1.22Z_MEM_ERROR . . . . .	4434

7.662.1.23Z_NEED_DICT . . . . .	4434
7.662.1.24Z_NO_COMPRESSION . . . . .	4434
7.662.1.25Z_NO_FLUSH . . . . .	4434
7.662.1.26Z_NULL . . . . .	4434
7.662.1.27Z_OK . . . . .	4434
7.662.1.28Z_PARTIAL_FLUSH . . . . .	4434
7.662.1.29Z_RLE . . . . .	4434
7.662.1.30Z_STREAM_END . . . . .	4434
7.662.1.31Z_STREAM_ERROR . . . . .	4434
7.662.1.32Z_SYNC_FLUSH . . . . .	4434
7.662.1.33Z_TEXT . . . . .	4434
7.662.1.34Z_TREES . . . . .	4434
7.662.1.35Z_UNKNOWN . . . . .	4434
7.662.1.36Z_VERSION_ERROR . . . . .	4434
7.662.1.37ZLIB_VER_MAJOR . . . . .	4434
7.662.1.38ZLIB_VER_MINOR . . . . .	4434
7.662.1.39ZLIB_VER_REVISION . . . . .	4434
7.662.1.40ZLIB_VER_SUBREVISION . . . . .	4434
7.662.1.41ZLIB_VERNUM . . . . .	4434
7.662.1.42zlib_version . . . . .	4435
7.662.1.43ZLIB_VERSION . . . . .	4435
7.662.2 Typedef Documentation . . . . .	4435
7.662.2.1 gz_header . . . . .	4435
7.662.2.2 gz_headerp . . . . .	4435
7.662.2.3 gzFile . . . . .	4435
7.662.2.4 OF . . . . .	4435
7.662.2.5 z_stream . . . . .	4435
7.662.2.6 z_streamp . . . . .	4435
7.662.3 Function Documentation . . . . .	4435
7.662.3.1 OF . . . . .	4435
7.662.3.2 OF . . . . .	4435
7.662.3.3 OF . . . . .	4435
7.662.3.4 OF . . . . .	4435
7.662.3.5 OF . . . . .	4435

7.662.3.6 OF . . . . .	4435
7.662.3.7 OF . . . . .	4435
7.662.3.8 OF . . . . .	4435
7.662.3.9 OF . . . . .	4435
7.662.3.10OF . . . . .	4435
7.662.3.11OF . . . . .	4435
7.662.3.12OF . . . . .	4436
7.662.3.13OF . . . . .	4436
7.662.3.14OF . . . . .	4436
7.662.3.15OF . . . . .	4436
7.662.3.16OF . . . . .	4436
7.662.3.17OF . . . . .	4436
7.662.3.18OF . . . . .	4436
7.662.3.19OF . . . . .	4436
7.662.3.20OF . . . . .	4436
7.662.3.21OF . . . . .	4436
7.662.3.22OF . . . . .	4436
7.662.3.23OF . . . . .	4436
7.662.3.24OF . . . . .	4436
7.662.3.25OF . . . . .	4436
7.662.3.26OF . . . . .	4436
7.662.3.27OF . . . . .	4436
7.662.3.28OF . . . . .	4436
7.662.3.29OF . . . . .	4436
7.662.3.30OF . . . . .	4436
7.662.3.31OF . . . . .	4436
7.662.3.32OF . . . . .	4436
7.662.3.33OF . . . . .	4437
7.662.3.34OF . . . . .	4437
7.662.3.35OF . . . . .	4437
7.662.3.36OF . . . . .	4437
7.662.3.37OF . . . . .	4437
7.662.3.38OF . . . . .	4437
7.662.3.39OF . . . . .	4437

7.662.3.40OF . . . . .	4437
7.662.3.41OF . . . . .	4437
7.662.3.42OF . . . . .	4437
7.663src/main/decaf/internal/util/zip/zutil.h File Reference . . . . .	4437
7.663.1 Define Documentation . . . . .	4438
7.663.1.1 Assert . . . . .	4438
7.663.1.2 DEF_MEM_LEVEL . . . . .	4438
7.663.1.3 DEF_WBITS . . . . .	4438
7.663.1.4 DYN_TREES . . . . .	4438
7.663.1.5 ERR_MSG . . . . .	4439
7.663.1.6 ERR_RETURN . . . . .	4439
7.663.1.7 F_OPEN . . . . .	4439
7.663.1.8 local . . . . .	4439
7.663.1.9 MAX_MATCH . . . . .	4439
7.663.1.10MIN_MATCH . . . . .	4439
7.663.1.11OS_CODE . . . . .	4439
7.663.1.12PRESET_DICT . . . . .	4439
7.663.1.13STATIC_TREES . . . . .	4439
7.663.1.14STORED_BLOCK . . . . .	4439
7.663.1.15Trace . . . . .	4439
7.663.1.16Tracec . . . . .	4439
7.663.1.17Tracecv . . . . .	4439
7.663.1.18Tracev . . . . .	4439
7.663.1.19Tracevv . . . . .	4439
7.663.1.20TRY_FREE . . . . .	4439
7.663.1.21ZALLOC . . . . .	4439
7.663.1.22ZFREE . . . . .	4439
7.663.1.23ZLIB_INTERNAL . . . . .	4439
7.663.2 Typedef Documentation . . . . .	4439
7.663.2.1 uch . . . . .	4439
7.663.2.2 uchf . . . . .	4439
7.663.2.3 ulg . . . . .	4439
7.663.2.4 ush . . . . .	4440
7.663.2.5 ushf . . . . .	4440

7.663.3 Function Documentation . . . . .	4440
7.663.3.1 OF . . . . .	4440
7.663.3.2 OF . . . . .	4440
7.663.3.3 OF . . . . .	4440
7.663.3.4 OF . . . . .	4440
7.663.3.5 OF . . . . .	4440
7.663.3.6 OF . . . . .	4440
7.663.4 Variable Documentation . . . . .	4440
7.663.4.1 z_errmsg . . . . .	4440
7.664src/main/decaf/io/BlockingByteArrayInputStream.h File Reference . . . .	4440
7.665src/main/decaf/io/BufferedInputStream.h File Reference . . . . .	4441
7.666src/main/decaf/io/BufferedOutputStream.h File Reference . . . . .	4441
7.667src/main/decaf/io/ByteArrayInputStream.h File Reference . . . . .	4442
7.668src/main/decaf/io/ByteArrayOutputStream.h File Reference . . . . .	4442
7.669src/main/decaf/io/DataInput.h File Reference . . . . .	4443
7.670src/main/decaf/io/DataInputStream.h File Reference . . . . .	4443
7.671src/main/decaf/io/DataOutput.h File Reference . . . . .	4444
7.672src/main/decaf/io/DataOutputStream.h File Reference . . . . .	4444
7.673src/main/decaf/io/EOFException.h File Reference . . . . .	4445
7.674src/main/decaf/io/FileDescriptor.h File Reference . . . . .	4445
7.675src/main/decaf/io/FilterInputStream.h File Reference . . . . .	4446
7.676src/main/decaf/io/FilterOutputStream.h File Reference . . . . .	4446
7.677src/main/decaf/io/Flushable.h File Reference . . . . .	4447
7.678src/main/decaf/io/InputStream.h File Reference . . . . .	4447
7.679src/main/decaf/io/InputStreamReader.h File Reference . . . . .	4448
7.680src/main/decaf/io/InterruptedIOException.h File Reference . . . . .	4448
7.681src/main/decaf/io/IOException.h File Reference . . . . .	4449
7.682src/main/decaf/io/OutputStream.h File Reference . . . . .	4449
7.683src/main/decaf/io/OutputStreamWriter.h File Reference . . . . .	4450
7.684src/main/decaf/io/PushbackInputStream.h File Reference . . . . .	4450
7.685src/main/decaf/io/Reader.h File Reference . . . . .	4450
7.686src/main/decaf/io/UnsupportedEncodingException.h File Reference . . .	4451
7.687src/main/decaf/io/UTFDataFormatException.h File Reference . . . . .	4451
7.688src/main/decaf/io/Writer.h File Reference . . . . .	4452

7.689src/main/decaf/lang/Appendable.h File Reference . . . . .	4452
7.690src/main/decaf/lang/ArrayPointer.h File Reference . . . . .	4453
7.691src/main/decaf/lang/Boolean.h File Reference . . . . .	4454
7.692src/main/decaf/lang/Byte.h File Reference . . . . .	4454
7.693src/main/decaf/lang/Character.h File Reference . . . . .	4455
7.694src/main/decaf/lang/CharSequence.h File Reference . . . . .	4455
7.695src/main/decaf/lang/Comparable.h File Reference . . . . .	4456
7.696src/main/decaf/lang/Double.h File Reference . . . . .	4456
7.697src/main/decaf/lang/Exception.h File Reference . . . . .	4457
7.698src/main/decaf/lang/exceptions/ClassCastException.h File Reference . .	4457
7.699src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference	4458
7.700src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Ref- erence . . . . .	4458
7.701src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Ref- erence . . . . .	4459
7.702src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Ref- erence . . . . .	4459
7.703src/main/decaf/lang/exceptions/InterruptedException.h File Reference .	4459
7.704src/main/decaf/lang/exceptions/InvalidStateException.h File Reference .	4460
7.705src/main/decaf/lang/exceptions/NoSuchElementException.h File Refer- ence . . . . .	4460
7.706src/main/decaf/lang/exceptions/NullPointerException.h File Reference .	4461
7.707src/main/decaf/lang/exceptions/NumberFormatException.h File Reference	4461
7.708src/main/decaf/lang/exceptions/RuntimeException.h File Reference . . .	4462
7.709src/main/decaf/lang/Float.h File Reference . . . . .	4462
7.710src/main/decaf/lang/Integer.h File Reference . . . . .	4462
7.711src/main/decaf/lang/Iterable.h File Reference . . . . .	4463
7.712src/main/decaf/lang/Long.h File Reference . . . . .	4463
7.713src/main/decaf/lang/Math.h File Reference . . . . .	4464
7.714src/main/decaf/lang/Number.h File Reference . . . . .	4464
7.715src/main/decaf/lang/Pointer.h File Reference . . . . .	4465
7.716src/main/decaf/lang/Readable.h File Reference . . . . .	4466
7.717src/main/decaf/lang/Runnable.h File Reference . . . . .	4466
7.718src/main/decaf/lang/Runtime.h File Reference . . . . .	4467
7.719src/main/decaf/lang/Short.h File Reference . . . . .	4467



7.720src/main/decaf/lang/String.h File Reference . . . . .	4468
7.721src/main/decaf/lang/System.h File Reference . . . . .	4468
7.722src/main/decaf/lang/Thread.h File Reference . . . . .	4469
7.723src/main/decaf/lang/ThreadGroup.h File Reference . . . . .	4469
7.724src/main/decaf/lang/Throwable.h File Reference . . . . .	4470
7.725src/main/decaf/net/BindException.h File Reference . . . . .	4470
7.726src/main/decaf/net/ConnectException.h File Reference . . . . .	4471
7.727src/main/decaf/net/HttpRetryException.h File Reference . . . . .	4471
7.728src/main/decaf/net/Inet4Address.h File Reference . . . . .	4471
7.729src/main/decaf/net/Inet6Address.h File Reference . . . . .	4472
7.730src/main/decaf/net/InetAddress.h File Reference . . . . .	4472
7.731src/main/decaf/net/InetSocketAddress.h File Reference . . . . .	4473
7.732src/main/decaf/net/MalformedURLException.h File Reference . . . . .	4473
7.733src/main/decaf/net/NoRouteToHostException.h File Reference . . . . .	4474
7.734src/main/decaf/net/PortUnreachableException.h File Reference . . . . .	4474
7.735src/main/decaf/net/ProtocolException.h File Reference . . . . .	4474
7.736src/main/decaf/net/ServerSocket.h File Reference . . . . .	4475
7.737src/main/decaf/net/ServerSocketFactory.h File Reference . . . . .	4475
7.738src/main/decaf/net/Socket.h File Reference . . . . .	4476
7.739src/main/decaf/net/SocketAddress.h File Reference . . . . .	4477
7.740src/main/decaf/net/SocketError.h File Reference . . . . .	4477
7.741src/main/decaf/net/SocketException.h File Reference . . . . .	4477
7.742src/main/decaf/net/SocketFactory.h File Reference . . . . .	4478
7.743src/main/decaf/net/SocketImpl.h File Reference . . . . .	4478
7.744src/main/decaf/net/SocketImplFactory.h File Reference . . . . .	4479
7.745src/main/decaf/net/SocketOptions.h File Reference . . . . .	4479
7.746src/main/decaf/net/SocketTimeoutException.h File Reference . . . . .	4480
7.747src/main/decaf/net/ssl/SSLContext.h File Reference . . . . .	4480
7.748src/main/decaf/net/ssl/SSLContextSpi.h File Reference . . . . .	4480
7.749src/main/decaf/net/ssl/SSLParameters.h File Reference . . . . .	4481
7.750src/main/decaf/net/ssl/SSLServerSocket.h File Reference . . . . .	4481
7.751src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference . . . . .	4482
7.752src/main/decaf/net/ssl/SSLSocket.h File Reference . . . . .	4482
7.753src/main/decaf/net/ssl/SSLSocketFactory.h File Reference . . . . .	4483

7.754src/main/decaf/net/UnknownHostException.h File Reference . . . . .	4483
7.755src/main/decaf/net/UnknownServiceException.h File Reference . . . . .	4484
7.756src/main/decaf/net/URI.h File Reference . . . . .	4484
7.757src/main/decaf/net/URISyntaxException.h File Reference . . . . .	4485
7.758src/main/decaf/net/URL.h File Reference . . . . .	4485
7.759src/main/decaf/net/URLDecoder.h File Reference . . . . .	4486
7.760src/main/decaf/net/URLEncoder.h File Reference . . . . .	4486
7.761src/main/decaf/nio/Buffer.h File Reference . . . . .	4486
7.762src/main/decaf/nio/BufferOverflowException.h File Reference . . . . .	4487
7.763src/main/decaf/nio/BufferUnderflowException.h File Reference . . . . .	4487
7.764src/main/decaf/nio/ByteBuffer.h File Reference . . . . .	4488
7.765src/main/decaf/nio/CharBuffer.h File Reference . . . . .	4488
7.766src/main/decaf/nio/DoubleBuffer.h File Reference . . . . .	4489
7.767src/main/decaf/nio/FloatBuffer.h File Reference . . . . .	4489
7.768src/main/decaf/nio/IntBuffer.h File Reference . . . . .	4490
7.769src/main/decaf/nio/InvalidMarkException.h File Reference . . . . .	4490
7.770src/main/decaf/nio/LongBuffer.h File Reference . . . . .	4491
7.771src/main/decaf/nio/ReadOnlyBufferException.h File Reference . . . . .	4491
7.772src/main/decaf/nio/ShortBuffer.h File Reference . . . . .	4492
7.773src/main/decaf/security/auth/x500/X500Principal.h File Reference . . . . .	4492
7.774src/main/decaf/security/cert/Certificate.h File Reference . . . . .	4493
7.775src/main/decaf/security/cert/CertificateEncodingException.h File Reference . . . . .	4494
7.776src/main/decaf/security/cert/CertificateException.h File Reference . . . . .	4494
7.777src/main/decaf/security/cert/CertificateExpiredException.h File Reference . . . . .	4494
7.778src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference . . . . .	4495
7.779src/main/decaf/security/cert/CertificateParsingException.h File Reference . . . . .	4495
7.780src/main/decaf/security/cert/X509Certificate.h File Reference . . . . .	4496
7.781src/main/decaf/security/GeneralSecurityException.h File Reference . . . . .	4496
7.782src/main/decaf/security/InvalidKeyException.h File Reference . . . . .	4497
7.783src/main/decaf/security/Key.h File Reference . . . . .	4497
7.784src/main/decaf/security/KeyException.h File Reference . . . . .	4498
7.785src/main/decaf/security/KeyManagementException.h File Reference . . . . .	4498

7.786src/main/decaf/security/NoSuchAlgorithmException.h File Reference . . .	4498
7.787src/main/decaf/security/NoSuchProviderException.h File Reference . . .	4499
7.788src/main/decaf/security/Principal.h File Reference . . . . .	4499
7.789src/main/decaf/security/PublicKey.h File Reference . . . . .	4500
7.790src/main/decaf/security/SecureRandom.h File Reference . . . . .	4500
7.791src/main/decaf/security/SecureRandomSpi.h File Reference . . . . .	4501
7.792src/main/decaf/security/SignatureException.h File Reference . . . . .	4501
7.793src/main/decaf/util/AbstractCollection.h File Reference . . . . .	4501
7.794src/main/decaf/util/AbstractList.h File Reference . . . . .	4502
7.795src/main/decaf/util/AbstractMap.h File Reference . . . . .	4503
7.796src/main/decaf/util/AbstractQueue.h File Reference . . . . .	4503
7.797src/main/decaf/util/AbstractSequentialList.h File Reference . . . . .	4504
7.798src/main/decaf/util/AbstractSet.h File Reference . . . . .	4505
7.799src/main/decaf/util/Collection.h File Reference . . . . .	4505
7.800src/main/decaf/util/Comparator.h File Reference . . . . .	4506
7.801src/main/decaf/util/comparators/Less.h File Reference . . . . .	4506
7.802src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference .	4507
7.803src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference . .	4507
7.804src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference	4508
7.805src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference	4508
7.806src/main/decaf/util/concurrent/BlockingQueue.h File Reference . . . . .	4509
7.807src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference .	4509
7.808src/main/decaf/util/concurrent/Callable.h File Reference . . . . .	4510
7.809src/main/decaf/util/concurrent/CancellationException.h File Reference .	4510
7.810src/main/decaf/util/concurrent/Concurrent.h File Reference . . . . .	4511
7.810.1 Define Documentation . . . . .	4511
7.810.1.1 synchronized . . . . .	4511
7.810.1.2 WAIT_INFINITE . . . . .	4511
7.811src/main/decaf/util/concurrent/ConcurrentMap.h File Reference . . . . .	4512
7.812src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference . . . .	4512
7.813src/main/decaf/util/concurrent/CountDownLatch.h File Reference . . . .	4513
7.814src/main/decaf/util/concurrent/Delayed.h File Reference . . . . .	4513
7.815src/main/decaf/util/concurrent/ExecutionException.h File Reference . . .	4514
7.816src/main/decaf/util/concurrent/Executor.h File Reference . . . . .	4514

7.817src/main/decaf/util/concurrent/ExecutorService.h File Reference . . . . .	4515
7.818src/main/decaf/util/concurrent/Future.h File Reference . . . . .	4515
7.819src/main/decaf/util/concurrent/Lock.h File Reference . . . . .	4516
7.820src/main/decaf/util/concurrent/locks/Lock.h File Reference . . . . .	4516
7.821src/main/decaf/util/concurrent/locks/Condition.h File Reference . . . . .	4517
7.822src/main/decaf/util/concurrent/locks/LockSupport.h File Reference . . . . .	4517
7.823src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference . . . . .	4518
7.824src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference . . . . .	4518
7.825src/main/decaf/util/concurrent/Mutex.h File Reference . . . . .	4519
7.826src/main/decaf/util/concurrent/PooledThread.h File Reference . . . . .	4519
7.827src/main/decaf/util/concurrent/PooledThreadListener.h File Reference . . . . .	4520
7.828src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference . . . . .	4520
7.829src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference . . . . .	4521
7.830src/main/decaf/util/concurrent/Semaphore.h File Reference . . . . .	4521
7.831src/main/decaf/util/concurrent/Synchronizable.h File Reference . . . . .	4522
7.832src/main/decaf/util/concurrent/SynchronousQueue.h File Reference . . . . .	4522
7.833src/main/decaf/util/concurrent/TaskListener.h File Reference . . . . .	4523
7.834src/main/decaf/util/concurrent/ThreadFactory.h File Reference . . . . .	4523
7.835src/main/decaf/util/concurrent/ThreadPool.h File Reference . . . . .	4524
7.836src/main/decaf/util/concurrent/TimeoutException.h File Reference . . . . .	4524
7.837src/main/decaf/util/concurrent/TimeUnit.h File Reference . . . . .	4525
7.838src/main/decaf/util/Date.h File Reference . . . . .	4526
7.839src/main/decaf/util/Iterator.h File Reference . . . . .	4526
7.840src/main/decaf/util/List.h File Reference . . . . .	4526
7.841src/main/decaf/util/ListIterator.h File Reference . . . . .	4527
7.842src/main/decaf/util/logging/ConsoleHandler.h File Reference . . . . .	4528
7.843src/main/decaf/util/logging/ErrorHandler.h File Reference . . . . .	4528
7.844src/main/decaf/util/logging/Filter.h File Reference . . . . .	4529
7.845src/main/decaf/util/logging/Formatter.h File Reference . . . . .	4529
7.846src/main/decaf/util/logging/Handler.h File Reference . . . . .	4529
7.847src/main/decaf/util/logging/Level.h File Reference . . . . .	4530
7.848src/main/decaf/util/logging/Logger.h File Reference . . . . .	4531
7.849src/main/decaf/util/logging/LoggerCommon.h File Reference . . . . .	4531

7.850src/main/decaf/util/logging/LoggerDefines.h File Reference . . . . .	4532
7.850.1 Define Documentation . . . . .	4532
7.850.1.1 LOGDECAF_DEBUG . . . . .	4532
7.850.1.2 LOGDECAF_DEBUG_1 . . . . .	4532
7.850.1.3 LOGDECAF_DECLARE . . . . .	4533
7.850.1.4 LOGDECAF_DECLARE_LOCAL . . . . .	4533
7.850.1.5 LOGDECAF_ERROR . . . . .	4533
7.850.1.6 LOGDECAF_FATAL . . . . .	4533
7.850.1.7 LOGDECAF_INFO . . . . .	4533
7.850.1.8 LOGDECAF_INITIALIZE . . . . .	4533
7.850.1.9 LOGDECAF_WARN . . . . .	4533
7.851src/main/decaf/util/logging/LoggerHierarchy.h File Reference . . . . .	4533
7.852src/main/decaf/util/logging/LogManager.h File Reference . . . . .	4533
7.853src/main/decaf/util/logging/LogRecord.h File Reference . . . . .	4534
7.854src/main/decaf/util/logging/LogWriter.h File Reference . . . . .	4535
7.855src/main/decaf/util/logging/MarkBlockLogger.h File Reference . . . . .	4535
7.856src/main/decaf/util/logging/PropertiesChangeListener.h File Reference . . . . .	4536
7.857src/main/decaf/util/logging/SimpleFormatter.h File Reference . . . . .	4536
7.858src/main/decaf/util/logging/SimpleLogger.h File Reference . . . . .	4537
7.859src/main/decaf/util/logging/StreamHandler.h File Reference . . . . .	4537
7.860src/main/decaf/util/logging/XMLFormatter.h File Reference . . . . .	4538
7.861src/main/decaf/util/Map.h File Reference . . . . .	4538
7.862src/main/decaf/util/PriorityQueue.h File Reference . . . . .	4539
7.863src/main/decaf/util/Properties.h File Reference . . . . .	4539
7.864src/main/decaf/util/Random.h File Reference . . . . .	4540
7.865src/main/decaf/util/Set.h File Reference . . . . .	4541
7.866src/main/decaf/util/StList.h File Reference . . . . .	4541
7.867src/main/decaf/util/StMap.h File Reference . . . . .	4542
7.868src/main/decaf/util/StQueue.h File Reference . . . . .	4542
7.869src/main/decaf/util/StSet.h File Reference . . . . .	4543
7.870src/main/decaf/util/StringTokenizer.h File Reference . . . . .	4544
7.871src/main/decaf/util/Timer.h File Reference . . . . .	4544
7.872src/main/decaf/util/TimerTask.h File Reference . . . . .	4545
7.873src/main/decaf/util/UUID.h File Reference . . . . .	4545

7.874src/main/decaf/util/zip/Adler32.h File Reference . . . . .	4546
7.875src/main/decaf/util/zip/CheckedInputStream.h File Reference . . . . .	4546
7.876src/main/decaf/util/zip/CheckedOutputStream.h File Reference . . . . .	4547
7.877src/main/decaf/util/zip/Checksum.h File Reference . . . . .	4547
7.878src/main/decaf/util/zip/CRC32.h File Reference . . . . .	4548
7.879src/main/decaf/util/zip/DataFormatException.h File Reference . . . . .	4548
7.880src/main/decaf/util/zip/Deflater.h File Reference . . . . .	4549
7.881src/main/decaf/util/zip/DeflaterOutputStream.h File Reference . . . . .	4549
7.882src/main/decaf/util/zip/Inflater.h File Reference . . . . .	4550
7.883src/main/decaf/util/zip/InflaterInputStream.h File Reference . . . . .	4550
7.884src/main/decaf/util/zip/ZipException.h File Reference . . . . .	4551

## Chapter 1

# Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>activemq</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	93
<b>activemq::cmsutil</b> . . . . .	94
<b>activemq::commands</b> . . . . .	95
<b>activemq::core</b> . . . . .	96
<b>activemq::core::policies</b> . . . . .	97
<b>activemq::exceptions</b> . . . . .	97
<b>activemq::io</b> . . . . .	98
<b>activemq::library</b> . . . . .	98
<b>activemq::state</b> . . . . .	98
<b>activemq::threads</b> . . . . .	98
<b>activemq::transport</b> . . . . .	99
<b>activemq::transport::correlator</b> . . . . .	100
<b>activemq::transport::failover</b> . . . . .	100
<b>activemq::transport::inactivity</b> . . . . .	100
<b>activemq::transport::logging</b> . . . . .	100
<b>activemq::transport::mock</b> . . . . .	101
<b>activemq::transport::tcp</b> . . . . .	101
<b>activemq::util</b> . . . . .	101
<b>activemq::wireformat</b> . . . . .	102
<b>activemq::wireformat::openwire</b> . . . . .	102
<b>activemq::wireformat::openwire::marshal</b> . . . . .	103
<b>activemq::wireformat::openwire::marshal::v1</b> . . . . .	103
<b>activemq::wireformat::openwire::marshal::v2</b> . . . . .	106
<b>activemq::wireformat::openwire::marshal::v3</b> . . . . .	110
<b>activemq::wireformat::openwire::marshal::v4</b> . . . . .	113
<b>activemq::wireformat::openwire::marshal::v5</b> . . . . .	116
<b>activemq::wireformat::openwire::marshal::v6</b> . . . . .	119
<b>activemq::wireformat::openwire::utils</b> . . . . .	122

<b>activemq::wireformat::stomp</b> . . . . .	122
<b>cms</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	122
<b>decaf</b> (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements ) . . . . .	125
<b>decaf::internal</b> . . . . .	125
<b>decaf::internal::io</b> . . . . .	126
<b>decaf::internal::net</b> . . . . .	126
<b>decaf::internal::net::ssl</b> . . . . .	127
<b>decaf::internal::net::ssl::openssl</b> . . . . .	127
<b>decaf::internal::net::tcp</b> . . . . .	128
<b>decaf::internal::nio</b> . . . . .	128
<b>decaf::internal::security</b> . . . . .	129
<b>decaf::internal::util</b> . . . . .	129
<b>decaf::internal::util::concurrent</b> . . . . .	129
<b>decaf::io</b> . . . . .	130
<b>decaf::lang</b> . . . . .	131
<b>decaf::lang::exceptions</b> . . . . .	134
<b>decaf::net</b> . . . . .	134
<b>decaf::net::ssl</b> . . . . .	136
<b>decaf::nio</b> . . . . .	136
<b>decaf::security</b> . . . . .	137
<b>decaf::security::auth</b> . . . . .	137
<b>decaf::security::auth::x500</b> . . . . .	137
<b>decaf::security::cert</b> . . . . .	138
<b>decaf::util</b> . . . . .	138
<b>decaf::util::comparators</b> . . . . .	140
<b>decaf::util::concurrent</b> . . . . .	140
<b>decaf::util::concurrent::atomic</b> . . . . .	141
<b>decaf::util::concurrent::locks</b> . . . . .	142
<b>decaf::util::logging</b> . . . . .	142
<b>decaf::util::zip</b> . . . . .	144
<b>std</b> . . . . .	145



## Chapter 2

# Data Structure Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

activemq::core::ActiveMQAckHandler . . . . .	171
activemq::core::ActiveMQConstants . . . . .	279
activemq::library::ActiveMQCPP . . . . .	292
activemq::core::ActiveMQTransactionContext . . . . .	688
decaf::lang::Appendable . . . . .	693
decaf::io::Writer . . . . .	3951
decaf::io::OutputStreamWriter . . . . .	2864
decaf::nio::CharBuffer . . . . .	1089
decaf::internal::nio::CharArrayBuffer . . . . .	1077
decaf::internal::AprPool . . . . .	696
decaf::lang::ArrayPointer< T, REFCOUNTER > . . . . .	697
decaf::util::concurrent::atomic::AtomicBoolean . . . . .	705
decaf::util::concurrent::atomic::AtomicRefCounter . . . . .	713
decaf::util::concurrent::atomic::AtomicReference< T > . . . . .	716
binary_function . . . . .	797
decaf::util::Comparator< ArrayPointer< T, R > > . . . . .	1189
decaf::lang::ArrayPointerComparator< T, R > . . . . .	704
decaf::util::Comparator< E > . . . . .	1189
decaf::util::comparators::Less< E > . . . . .	2287
decaf::util::Comparator< Pointer< T, R > > . . . . .	1189
decaf::lang::PointerComparator< T, R > . . . . .	2903
decaf::util::Comparator< T > . . . . .	1189
std::less< decaf::lang::ArrayPointer< T > > . . . . .	2289
std::less< decaf::lang::Pointer< T > > . . . . .	2289
activemq::wireformat::openwire::utils::BooleanStream . . . . .	818
decaf::nio::Buffer . . . . .	887
decaf::nio::ByteBuffer . . . . .	995
decaf::internal::nio::ByteBuffer . . . . .	951

decaf::nio::CharBuffer . . . . .	1089
decaf::nio::DoubleBuffer . . . . .	1773
decaf::internal::nio::DoubleArrayBuffer . . . . .	1762
decaf::nio::FloatBuffer . . . . .	1887
decaf::internal::nio::FloatArrayBuffer . . . . .	1876
decaf::nio::IntBuffer . . . . .	2026
decaf::internal::nio::IntArrayBuffer . . . . .	2015
decaf::nio::LongBuffer . . . . .	2403
decaf::internal::nio::LongArrayBuffer . . . . .	2392
decaf::nio::ShortBuffer . . . . .	3401
decaf::internal::nio::ShortArrayBuffer . . . . .	3390
decaf::internal::nio::BufferFactory . . . . .	901
decaf::internal::util::ByteArrayAdapter . . . . .	928
decaf::util::concurrent::Callable< V > . . . . .	1051
decaf::security::cert::Certificate . . . . .	1055
decaf::security::cert::X509Certificate . . . . .	3958
decaf::lang::CharSequence . . . . .	1107
decaf::lang::String . . . . .	3610
decaf::nio::CharBuffer . . . . .	1089
decaf::util::zip::Checksum . . . . .	1114
decaf::util::zip::Adler32 . . . . .	691
decaf::util::zip::CRC32 . . . . .	1490
cms::Closeable . . . . .	1119
activemq::commands::ActiveMQTempDestination . . . . .	547
activemq::commands::ActiveMQTempQueue . . . . .	574
activemq::commands::ActiveMQTempTopic . . . . .	602
cms::Connection . . . . .	1232
activemq::core::ActiveMQConnection . . . . .	244
cms::MessageConsumer . . . . .	2550
activemq::cmsutil::CachedConsumer . . . . .	1041
activemq::core::ActiveMQConsumer . . . . .	282
cms::MessageProducer . . . . .	2681
activemq::cmsutil::CachedProducer . . . . .	1044
activemq::core::ActiveMQProducer . . . . .	441
cms::QueueBrowser . . . . .	3098
activemq::core::ActiveMQQueueBrowser . . . . .	457
cms::Session . . . . .	3305
activemq::cmsutil::PooledSession . . . . .	2904
activemq::core::ActiveMQSession . . . . .	484
decaf::io::Closeable . . . . .	1120
activemq::transport::Transport . . . . .	3819
activemq::transport::CompositeTransport . . . . .	1197
activemq::transport::failover::FailoverTransport . . . . .	1835
activemq::transport::IOTransport . . . . .	2105
activemq::transport::mock::MockTransport . . . . .	2724
activemq::transport::TransportFilter . . . . .	3827
activemq::transport::correlator::ResponseCorrelator . . . . .	3232

activemq::transport::inactivity::InactivityMonitor . . . . .	1964
activemq::transport::logging::LoggingTransport . . . . .	2360
activemq::transport::tcp::TcpTransport . . . . .	3696
activemq::transport::tcp::SslTransport . . . . .	3518
activemq::wireformat::WireFormatNegotiator . . . . .	3946
activemq::wireformat::openwire::OpenWireFormatNegotiator . . . . .	2851
decaf::io::InputStream . . . . .	2002
decaf::internal::io::StandardInputStream . . . . .	3524
decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream . . . . .	2832
decaf::internal::net::tcp::TcpSocketInputStream . . . . .	3691
decaf::io::BlockingByteArrayInputStream . . . . .	800
decaf::io::ByteArrayInputStream . . . . .	984
decaf::io::FilterInputStream . . . . .	1854
activemq::io::LoggingInputStream . . . . .	2358
decaf::io::BufferedInputStream . . . . .	893
decaf::io::DataInputStream . . . . .	1532
decaf::io::PushbackInputStream . . . . .	3086
decaf::util::zip::CheckedInputStream . . . . .	1109
decaf::util::zip::InflaterInputStream . . . . .	1994
decaf::io::OutputStream . . . . .	2856
decaf::internal::io::StandardErrorOutputStream . . . . .	3521
decaf::internal::io::StandardOutputStream . . . . .	3525
decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream . . . . .	2835
decaf::internal::net::tcp::TcpSocketOutputStream . . . . .	3694
decaf::io::ByteArrayOutputStream . . . . .	992
decaf::io::FilterOutputStream . . . . .	1861
activemq::io::LoggingOutputStream . . . . .	2359
decaf::io::BufferedOutputStream . . . . .	899
decaf::io::DataOutputStream . . . . .	1546
decaf::util::zip::CheckedOutputStream . . . . .	1112
decaf::util::zip::DeflaterOutputStream . . . . .	1682
decaf::io::Reader . . . . .	3108
decaf::io::InputStreamReader . . . . .	2013
decaf::io::Writer . . . . .	3951
decaf::net::Socket . . . . .	3445
decaf::net::ssl::SSLSocket . . . . .	3506
decaf::internal::net::ssl::openssl::OpenSSLSocket . . . . .	2808
decaf::util::logging::Handler . . . . .	1941
decaf::util::logging::StreamHandler . . . . .	3591
decaf::util::logging::ConsoleHandler . . . . .	1367
activemq::cmsutil::CmsAccessor . . . . .	1123
activemq::cmsutil::CmsDestinationAccessor . . . . .	1127
activemq::cmsutil::CmsTemplate . . . . .	1140
cms::CMSException . . . . .	1130
cms::CMSSecurityException . . . . .	1139
cms::IllegalStateException . . . . .	1958
cms::InvalidClientIdException . . . . .	2091
cms::InvalidDestinationException . . . . .	2093

cms::InvalidSelectorException . . . . .	2099
cms::MessageEOFException . . . . .	2621
cms::MessageFormatException . . . . .	2622
cms::MessageNotReadableException . . . . .	2679
cms::MessageNotWriteableException . . . . .	2680
cms::UnsupportedOperationException . . . . .	3852
activemq::util::CMSExceptionSupport . . . . .	1134
cms::CMSProperties . . . . .	1135
activemq::util::ActiveMQProperties . . . . .	449
code . . . . .	1154
activemq::state::CommandVisitor . . . . .	1171
activemq::state::CommandVisitorAdapter . . . . .	1179
activemq::state::ConnectionStateTracker . . . . .	1361
decaf::lang::Comparable< T > . . . . .	1186
decaf::lang::Comparable< bool > . . . . .	1186
decaf::lang::Boolean . . . . .	810
decaf::lang::Comparable< Boolean > . . . . .	1186
decaf::lang::Boolean . . . . .	810
decaf::lang::Comparable< BrokerId > . . . . .	1186
activemq::commands::BrokerId . . . . .	828
decaf::lang::Comparable< Byte > . . . . .	1186
decaf::lang::Byte . . . . .	918
decaf::lang::Comparable< ByteBuffer > . . . . .	1186
decaf::nio::ByteBuffer . . . . .	995
decaf::lang::Comparable< char > . . . . .	1186
decaf::lang::Character . . . . .	1069
decaf::lang::Comparable< Character > . . . . .	1186
decaf::lang::Character . . . . .	1069
decaf::lang::Comparable< CharBuffer > . . . . .	1186
decaf::nio::CharBuffer . . . . .	1089
decaf::lang::Comparable< ConnectionId > . . . . .	1186
activemq::commands::ConnectionId . . . . .	1297
decaf::lang::Comparable< ConsumerId > . . . . .	1186
activemq::commands::ConsumerId . . . . .	1398
decaf::lang::Comparable< Date > . . . . .	1186
decaf::util::Date . . . . .	1633
decaf::lang::Comparable< Delayed > . . . . .	1186
decaf::util::concurrent::Delayed . . . . .	1686
decaf::lang::Comparable< Double > . . . . .	1186
decaf::lang::Double . . . . .	1751
decaf::lang::Comparable< double > . . . . .	1186
decaf::lang::Double . . . . .	1751
decaf::lang::Comparable< DoubleBuffer > . . . . .	1186
decaf::nio::DoubleBuffer . . . . .	1773

decaf::lang::Comparable< float > . . . . .	1186
decaf::lang::Float . . . . .	1865
decaf::lang::Comparable< Float > . . . . .	1186
decaf::lang::Float . . . . .	1865
decaf::lang::Comparable< FloatBuffer > . . . . .	1186
decaf::nio::FloatBuffer . . . . .	1887
decaf::lang::Comparable< int > . . . . .	1186
decaf::lang::Integer . . . . .	2038
decaf::lang::Comparable< IntBuffer > . . . . .	1186
decaf::nio::IntBuffer . . . . .	2026
decaf::lang::Comparable< Integer > . . . . .	1186
decaf::lang::Integer . . . . .	2038
decaf::lang::Comparable< Level > . . . . .	1186
decaf::util::logging::Level . . . . .	2290
decaf::lang::Comparable< LocalTransactionId > . . . . .	1186
activemq::commands::LocalTransactionId . . . . .	2306
decaf::lang::Comparable< Long > . . . . .	1186
decaf::lang::Long . . . . .	2377
decaf::lang::Comparable< long long > . . . . .	1186
decaf::lang::Long . . . . .	2377
decaf::lang::Comparable< LongBuffer > . . . . .	1186
decaf::nio::LongBuffer . . . . .	2403
decaf::lang::Comparable< MessageId > . . . . .	1186
activemq::commands::MessageId . . . . .	2623
decaf::lang::Comparable< ProducerId > . . . . .	1186
activemq::commands::ProducerId . . . . .	3014
decaf::lang::Comparable< SessionId > . . . . .	1186
activemq::commands::SessionId . . . . .	3320
decaf::lang::Comparable< Short > . . . . .	1186
decaf::lang::Short . . . . .	3380
decaf::lang::Comparable< short > . . . . .	1186
decaf::lang::Short . . . . .	3380
decaf::lang::Comparable< ShortBuffer > . . . . .	1186
decaf::nio::ShortBuffer . . . . .	3401
decaf::lang::Comparable< TimeUnit > . . . . .	1186
decaf::util::concurrent::TimeUnit . . . . .	3748
decaf::lang::Comparable< TransactionId > . . . . .	1186
activemq::commands::TransactionId . . . . .	3759
activemq::commands::LocalTransactionId . . . . .	2306
activemq::commands::XATransactionId . . . . .	3960
decaf::lang::Comparable< unsigned char > . . . . .	1186
decaf::lang::Byte . . . . .	918
decaf::lang::Comparable< URI > . . . . .	1186

decaf::net::URI	3853
decaf::lang::Comparable< UUID >	1186
decaf::util::UUID	3900
decaf::lang::Comparable< XATransactionId >	1186
activemq::commands::XATransactionId	3960
activemq::util::CompositeData	1191
decaf::util::concurrent::locks::Condition	1220
decaf::util::concurrent::ConditionHandle	1226
decaf::internal::util::concurrent::ConditionImpl	1228
cms::ConnectionFactory	1294
activemq::core::ActiveMQConnectionFactory	264
cms::ConnectionMetaData	1355
activemq::core::ActiveMQConnectionMetaData	275
activemq::state::ConnectionState	1358
activemq::state::ConsumerState	1459
decaf::util::concurrent::CountDownLatch	1487
ct_data_s	1492
decaf::io::DataInput	1523
decaf::io::DataOutput	1541
activemq::wireformat::openwire::marshal::DataStreamMarshaller	1577
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	770
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	308
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller	464
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	555
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	582
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	615
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	672
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	743
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	871
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	1250
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	1282
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	1343
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	1386
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1447
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller	1475
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1708
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	1919
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	2249
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2542
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2582
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2611
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	2670
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	182
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	224
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	348
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller	375
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	421
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	527

```
    activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller644
    activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller2716
    activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller3008
    activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller3056
    activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller 3153
    activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller3169
    activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller3201
    activemq::wireformat::openwire::marshal::v1::ResponseMarshaller 3255
        activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller1508
        activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller1573
        activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller1825
        activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller2073
    activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller 3360
    activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller3424
    activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller3793
    activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller . . . 840
    activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller . 1313
    activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller . 1414
    activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller 1741
    activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller2139
    activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller2168
    activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller . 2190
    activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller2221
    activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller . . 2648
    activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller2769
    activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller2891
        activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller2283
    activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller . . 3039
    activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller . . . 3344
    activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller3624
    activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller . 3766
        activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller2330
        activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller3976
    activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller 3939
    activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller320
        activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller476
        activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller566
            activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller594
            activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller623
        activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller684
    activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller 764
        activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller 883
        activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller1262
        activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller1270
        activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller1330
        activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller1373
        activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller1434
        activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller1462
        activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller1696
```

activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller1907  
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller2233  
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller2530  
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller2566  
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller2599  
activemq::wireformat::openwire::marshal::v2::MessageMarshaller . 2661  
    activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller190  
    activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller240  
    activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller360  
    activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller387  
    activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller433  
    activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller539  
    activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller656  
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller2700  
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller2988  
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller3052  
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller 3141  
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller3178  
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller3205  
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller 3241  
    activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller1496  
    activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller1561  
    activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller1809  
    activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller2061  
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller 3368  
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller3420  
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller3809  
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller . . . 852  
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller . 1301  
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller . 1402  
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller 1729  
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller2123  
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller2152  
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller . 2174  
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller2205  
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller . . 2628  
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller2749  
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller2874  
    activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller2271  
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller . . 3019  
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller . . . 3324  
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller3640  
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller . 3770  
    activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller2314  
    activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller3968  
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller 3931  
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller304  
    activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller460  
    activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller551



```
    activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller578
    activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller607
    activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller664
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller 730
    activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller 862
    activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller1242
    activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller1274
    activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller1335
    activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller1378
    activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller1439
    activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller1467
    activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller1700
    activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller1911
    activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller2237
    activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller2534
    activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller2570
    activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller2603
    activemq::wireformat::openwire::marshal::v3::MessageMarshaller . 2657
        activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller177
        activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller220
        activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller344
        activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller371
        activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller416
        activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller523
        activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller635
    activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller2708
    activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller2996
    activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller3064
    activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller 3149
    activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller3174
    activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller3209
    activemq::wireformat::openwire::marshal::v3::ResponseMarshaller 3250
        activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller1500
        activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller1565
        activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller1813
        activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller2065
    activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller 3364
    activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller3432
    activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller3797
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller . . . 832
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller . 1305
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller . 1406
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller 1733
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller2131
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller2156
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller . 2178
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller2209
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller . . 2640
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller2761
```

```

activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller2883
  activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller2267
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller . . . 3027
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller . . . 3340
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller3620
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller . 3774
  activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller2318
  activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller3980
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller 3943
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller312
  activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller468
  activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller558
    activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller586
    activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller611
  activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller668
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller 737
  activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller 867
  activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller1246
  activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller1278
  activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller1339
  activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller1382
  activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller1443
  activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller1471
  activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller1704
  activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller1915
  activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller2241
  activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller2538
  activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller2578
  activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller2607
  activemq::wireformat::openwire::marshal::v4::MessageMarshaller . 2666
    activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller186
    activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller228
    activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller352
    activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller379
    activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller425
    activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller531
    activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller640
  activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller2712
  activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller2992
  activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller3047
  activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller 3161
  activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller3190
  activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller3197
  activemq::wireformat::openwire::marshal::v4::ResponseMarshaller 3236
    activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller1504
    activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller1569
    activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller1821
    activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller2069
  activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller 3372

```

```
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller3436
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller3805
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller . . . 836
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller . 1309
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller . 1410
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller 1737
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller2135
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller2164
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller . 2186
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller2217
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller . . 2632
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller2765
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller2887
    activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller2279
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller . . 3023
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller . . . 3328
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller3632
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller . 3778
    activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller2326
    activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller3972
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller 3935
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller316
    activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller472
    activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller562
        activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller590
        activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller619
    activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller676
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller 750
    activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller 875
    activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller1254
    activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller1286
    activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller1347
    activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller1390
    activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller1451
    activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller1479
    activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller1716
    activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller1923
    activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller2245
    activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller2546
    activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller2574
    activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller2616
    activemq::wireformat::openwire::marshal::v5::MessageMarshaller . 2653
        activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller194
        activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller232
        activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller356
        activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller383
        activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller429
        activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller535
        activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller648
```

```

activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller2704
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller3000
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller3060
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller 3157
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller3186
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller3217
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller 3246
    activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller1512
    activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller1553
    activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller1817
    activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller2077
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller 3356
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller3428
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller3789
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller . . . 844
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller . 1317
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller . 1418
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller 1745
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller2127
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller2148
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller . 2194
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller2213
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller . . 2636
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller2757
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller2878
    activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller2275
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller . . 3031
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller . . . 3336
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller3628
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller . 3763
    activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller2322
    activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller3984
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller 3923
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller324
    activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller480
    activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller570
        activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller598
        activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller627
    activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller680
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller 757
    activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller 879
    activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller1258
    activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller1290
    activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller1351
    activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller1394
    activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller1455
    activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller1483
    activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller1712
    activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller1903

```

activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller	2228
activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller	2526
activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller	2586
activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	2595
activemq::wireformat::openwire::marshal::v6::MessageMarshaller	2674
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller	198
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller	236
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	364
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	391
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	437
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller	543
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	652
activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller	2720
activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller	3004
activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller	3068
activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller	3145
activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	3182
activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller	3213
activemq::wireformat::openwire::marshal::v6::ResponseMarshaller	3260
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller	1516
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller	1557
activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller	1804
activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller	2057
activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller	3352
activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller	3416
activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller	3801
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller	848
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller	1321
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller	1422
activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller	1725
activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller	2119
activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller	2160
activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller	2182
activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller	2201
activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller	2644
activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller	2753
activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller	2870
activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller	2262
activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller	3035
activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller	3332
activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller	3636
activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller	3781
activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller	2310
activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller	3964
activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller	3927
decaf::internal::net::ssl::DefaultSSLContext	1657
decaf::util::zip::Deflater	1672
cms::DeliveryMode	1687
cms::Destination	1688

cms::Queue	3093
activemq::commands::ActiveMQQueue	453
cms::TemporaryQueue	3701
activemq::commands::ActiveMQTempQueue	574
cms::TemporaryTopic	3703
activemq::commands::ActiveMQTempTopic	602
cms::Topic	3757
activemq::commands::ActiveMQTopic	660
activemq::commands::ActiveMQDestination::DestinationFilter	1691
activemq::cmsutil::DestinationResolver	1720
activemq::cmsutil::DynamicDestinationResolver	1786
activemq::core::DispatchData	1749
activemq::core::Dispatcher	1750
activemq::core::ActiveMQConsumer	282
activemq::core::ActiveMQSession	484
decaf::lang::DYNAMIC_CAST_TOKEN	1786
decaf::util::Map< K, V, COMPARATOR >::Entry	1788
decaf::util::logging::ErrorManager	1792
cms::ExceptionListener	1801
decaf::util::concurrent::Executor	1831
decaf::util::concurrent::ExecutorService	1833
decaf::io::FileDescriptor	1850
decaf::internal::net::SocketFileDescriptor	3471
decaf::util::logging::Filter	1853
decaf::io::Flushable	1899
decaf::io::OutputStream	2856
decaf::io::Writer	3951
decaf::util::logging::Formatter	1927
decaf::util::logging::SimpleFormatter	3442
decaf::util::logging::XMLFormatter	3988
decaf::util::concurrent::Future< V >	1929
activemq::transport::correlator::FutureResponse	1932
gz_header_s	1938
gz_state	1939
decaf::internal::util::HexStringParser	1945
activemq::wireformat::openwire::utils::HexTable	1947
activemq::util::IdGenerator	1951
decaf::net::InetAddress	1974
decaf::net::Inet4Address	1970
decaf::net::Inet6Address	1973
inflate_state	1982
decaf::util::zip::Inflater	1985
internal_state	2081
decaf::lang::Iterable< E >	2112
decaf::util::Collection< E >	1155
decaf::util::AbstractCollection< E >	147
decaf::util::List< E >	2296

decaf::util::AbstractList< E > . . . . .	161
decaf::util::AbstractSequentialList< E > . . . . .	167
decaf::util::StlList< E > . . . . .	3529
decaf::util::Queue< E > . . . . .	3094
decaf::util::AbstractQueue< E > . . . . .	163
decaf::util::concurrent::BlockingQueue< E > . . . . .	804
decaf::util::concurrent::SynchronousQueue< E > . . . . .	3660
decaf::util::PriorityQueue< E > . . . . .	2975
decaf::util::Set< E > . . . . .	3379
decaf::util::AbstractSet< E > . . . . .	168
decaf::util::StlSet< E > . . . . .	3564
decaf::lang::Iterable< PrimitiveValueNode > . . . . .	2112
decaf::util::Collection< PrimitiveValueNode > . . . . .	1155
decaf::util::AbstractCollection< PrimitiveValueNode > . . . . .	147
decaf::util::List< PrimitiveValueNode > . . . . .	2296
decaf::util::StlList< PrimitiveValueNode > . . . . .	3529
activemq::util::PrimitiveList . . . . .	2929
decaf::util::Iterator< T > . . . . .	2114
decaf::util::Iterator< E > . . . . .	2114
decaf::util::ListIterator< E > . . . . .	2303
decaf::security::Key . . . . .	2253
decaf::security::PublicKey . . . . .	3086
decaf::util::concurrent::Lock . . . . .	2334
decaf::util::concurrent::locks::Lock . . . . .	2336
decaf::util::concurrent::locks::ReentrantLock . . . . .	3126
decaf::util::concurrent::locks::LockSupport . . . . .	2341
decaf::util::logging::Logger . . . . .	2345
decaf::util::logging::LoggerHierarchy . . . . .	2357
decaf::util::logging::LogManager . . . . .	2363
decaf::util::logging::LogRecord . . . . .	2370
decaf::util::logging::LogWriter . . . . .	2375
activemq::util::LongSequenceGenerator . . . . .	2415
decaf::util::logging::MarkBlockLogger . . . . .	2443
activemq::wireformat::MarshalAware . . . . .	2444
activemq::commands::DataStructure . . . . .	1628
activemq::commands::BaseDataStructure . . . . .	793
activemq::commands::ActiveMQDestination . . . . .	293
activemq::commands::ActiveMQQueue . . . . .	453
activemq::commands::ActiveMQTempDestination . . . . .	547
activemq::commands::ActiveMQTopic . . . . .	660
activemq::commands::BooleanExpression . . . . .	816
activemq::commands::BrokerId . . . . .	828
activemq::commands::Command . . . . .	1165
activemq::commands::BaseCommand . . . . .	723
activemq::commands::BrokerError . . . . .	823
activemq::commands::BrokerInfo . . . . .	856
activemq::commands::ConnectionControl . . . . .	1237

activemq::commands::ConnectionError . . . . .	1266
activemq::commands::ConnectionInfo . . . . .	1324
activemq::commands::ConsumerControl . . . . .	1369
activemq::commands::ConsumerInfo . . . . .	1426
activemq::commands::ControlCommand . . . . .	1459
activemq::commands::DestinationInfo . . . . .	1691
activemq::commands::FlushCommand . . . . .	1900
activemq::commands::KeepAliveInfo . . . . .	2225
activemq::commands::Message . . . . .	2475
activemq::commands::ActiveMQMessageTemplate< T > . . . . .	395
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage > . . . . .	395
activemq::commands::ActiveMQBytesMessage . . . . .	202
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage > . . . . .	395
activemq::commands::ActiveMQMapMessage . . . . .	330
activemq::commands::ActiveMQMessageTemplate< cms::Message > . . . . .	395
activemq::commands::ActiveMQBlobMessage . . . . .	172
activemq::commands::ActiveMQMessage . . . . .	368
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage > . . . . .	395
activemq::commands::ActiveMQObjectMessage . . . . .	414
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage > . . . . .	395
activemq::commands::ActiveMQStreamMessage . . . . .	506
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage > . . . . .	395
activemq::commands::ActiveMQTextMessage . . . . .	631
activemq::commands::MessageAck . . . . .	2521
activemq::commands::MessageDispatch . . . . .	2555
activemq::commands::MessageDispatchNotification . . . . .	2590
activemq::commands::MessagePull . . . . .	2695
activemq::commands::ProducerAck . . . . .	2984
activemq::commands::ProducerInfo . . . . .	3043
activemq::commands::RemoveInfo . . . . .	3137
activemq::commands::RemoveSubscriptionInfo . . . . .	3165
activemq::commands::ReplayCommand . . . . .	3194
activemq::commands::Response . . . . .	3227
activemq::commands::DataArrayResponse . . . . .	1493
activemq::commands::DataResponse . . . . .	1550
activemq::commands::ExceptionResponse . . . . .	1802
activemq::commands::IntegerResponse . . . . .	2054
activemq::state::Tracked . . . . .	3758
activemq::commands::SessionInfo . . . . .	3348
activemq::commands::ShutdownInfo . . . . .	3413
activemq::commands::TransactionInfo . . . . .	3785
activemq::commands::WireFormatInfo . . . . .	3912
activemq::commands::ConnectionId . . . . .	1297
activemq::commands::ConsumerId . . . . .	1398



activemq::commands::DiscoveryEvent . . . . .	1722
activemq::commands::JournalQueueAck . . . . .	2116
activemq::commands::JournalTopicAck . . . . .	2143
activemq::commands::JournalTrace . . . . .	2171
activemq::commands::JournalTransaction . . . . .	2198
activemq::commands::MessageId . . . . .	2623
activemq::commands::NetworkBridgeFilter . . . . .	2746
activemq::commands::PartialCommand . . . . .	2866
activemq::commands::LastPartialCommand . . . . .	2260
activemq::commands::ProducerId . . . . .	3014
activemq::commands::SessionId . . . . .	3320
activemq::commands::SubscriptionInfo . . . . .	3616
activemq::commands::TransactionId . . . . .	3759
activemq::wireformat::openwire::marshal::v6::MarshallerFactory . . . . .	2447
activemq::wireformat::openwire::marshal::v3::MarshallerFactory . . . . .	2447
activemq::wireformat::openwire::marshal::v4::MarshallerFactory . . . . .	2448
activemq::wireformat::openwire::marshal::v5::MarshallerFactory . . . . .	2449
activemq::wireformat::openwire::marshal::v1::MarshallerFactory . . . . .	2450
activemq::wireformat::openwire::marshal::v2::MarshallerFactory . . . . .	2450
activemq::util::MarshallingSupport . . . . .	2451
decaf::lang::Math . . . . .	2455
cms::Message . . . . .	2493
activemq::commands::ActiveMQMessageTemplate< cms::Message > . . . . .	395
cms::BytesMessage . . . . .	1023
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage	
> . . . . .	395
cms::MapMessage . . . . .	2431
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage	
> . . . . .	395
cms::ObjectMessage . . . . .	2791
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage	
> . . . . .	395
cms::StreamMessage . . . . .	3595
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage	
> . . . . .	395
cms::TextMessage . . . . .	3704
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage	
> . . . . .	395
activemq::cmsutil::MessageCreator . . . . .	2554
cms::MessageEnumeration . . . . .	2620
activemq::core::ActiveMQQueueBrowser . . . . .	457
cms::MessageListener . . . . .	2652
activemq::wireformat::openwire::utils::MessagePropertyInterceptor . . . . .	2689
decaf::util::concurrent::MutexHandle . . . . .	2741
decaf::internal::util::concurrent::MutexImpl . . . . .	2742
decaf::internal::net::Network . . . . .	2744
decaf::lang::Number . . . . .	2786
decaf::lang::Byte . . . . .	918

decaf::lang::Character . . . . .	1069
decaf::lang::Double . . . . .	1751
decaf::lang::Float . . . . .	1865
decaf::lang::Integer . . . . .	2038
decaf::lang::Long . . . . .	2377
decaf::lang::Short . . . . .	3380
decaf::util::concurrent::atomic::AtomicInteger . . . . .	708
decaf::internal::net::ssl::openssl::OpenSSLParameters . . . . .	2795
decaf::lang::Pointer< T, REFCOUNTER > . . . . .	2896
decaf::util::concurrent::PooledThreadListener . . . . .	2920
decaf::util::concurrent::ThreadPool . . . . .	3718
activemq::core::PrefetchPolicy . . . . .	2924
activemq::core::policies::DefaultPrefetchPolicy . . . . .	1640
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller . . . . .	2951
activemq::util::PrimitiveValueNode::PrimitiveValue . . . . .	2957
activemq::util::PrimitiveValueConverter . . . . .	2959
activemq::util::PrimitiveValueNode . . . . .	2960
decaf::security::Principal . . . . .	2974
decaf::security::auth::x500::X500Principal . . . . .	3957
activemq::cmsutil::ProducerCallback . . . . .	3012
activemq::cmsutil::CmsTemplate::SendExecutor . . . . .	3290
activemq::state::ProducerState . . . . .	3072
decaf::util::Properties . . . . .	3072
decaf::util::logging::PropertiesChangeListener . . . . .	3082
decaf::util::Random . . . . .	3100
decaf::security::SecureRandom . . . . .	3269
decaf::lang::Readable . . . . .	3106
decaf::io::Reader . . . . .	3108
decaf::util::concurrent::locks::ReadWriteLock . . . . .	3117
activemq::core::RedeliveryPolicy . . . . .	3121
activemq::core::policies::DefaultRedeliveryPolicy . . . . .	1644
decaf::util::concurrent::RejectedExecutionHandler . . . . .	3136
decaf::internal::util::Resource . . . . .	3223
decaf::internal::util::GenericResource< T > . . . . .	1937
decaf::internal::util::ResourceLifecycleManager . . . . .	3224
activemq::cmsutil::ResourceLifecycleManager . . . . .	3224
activemq::transport::mock::ResponseBuilder . . . . .	3231
activemq::wireformat::openwire::OpenWireResponseBuilder . . . . .	2854
decaf::lang::Runnable . . . . .	3264
activemq::threads::CompositeTaskRunner . . . . .	1194
activemq::threads::DedicatedTaskRunner . . . . .	1638
activemq::transport::IOTransport . . . . .	2105
decaf::lang::Thread . . . . .	3707
activemq::transport::mock::InternalCommandListener . . . . .	2085
decaf::util::concurrent::PooledThread . . . . .	2918
decaf::util::TimerTask . . . . .	3743

activemq::transport::inactivity::ReadChecker . . . . .	3107
activemq::transport::inactivity::WriteChecker . . . . .	3950
decaf::lang::Runtime . . . . .	3265
decaf::internal::DecafRuntime . . . . .	1637
decaf::security::SecureRandomSpi . . . . .	3278
decaf::internal::security::SecureRandomImpl . . . . .	3275
decaf::internal::security::SecureRandomImpl . . . . .	3275
decaf::util::concurrent::Semaphore . . . . .	3280
decaf::net::ServerSocket . . . . .	3292
decaf::net::ssl::SSLServerSocket . . . . .	3498
decaf::internal::net::ssl::openssl::OpenSSLServerSocket . . . . .	2797
decaf::net::ServerSocketFactory . . . . .	3301
decaf::internal::net::DefaultServerSocketFactory . . . . .	1648
decaf::net::ssl::SSLServerSocketFactory . . . . .	3504
decaf::internal::net::ssl::DefaultSSLServerSocketFactory . . . . .	1658
decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory . . . . .	2803
activemq::cmsutil::SessionCallback . . . . .	3319
activemq::cmsutil::CmsTemplate::ProducerExecutor . . . . .	3013
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor . . . . .	3221
activemq::cmsutil::CmsTemplate::ReceiveExecutor . . . . .	3119
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor . . . . .	3222
activemq::cmsutil::SessionPool . . . . .	3376
activemq::state::SessionState . . . . .	3378
decaf::util::logging::SimpleLogger . . . . .	3444
decaf::net::SocketAddress . . . . .	3463
decaf::net::InetSocketAddress . . . . .	1982
decaf::net::SocketError . . . . .	3464
decaf::net::SocketFactory . . . . .	3467
decaf::internal::net::DefaultSocketFactory . . . . .	1652
decaf::net::ssl::SSLSocketFactory . . . . .	3515
decaf::internal::net::ssl::DefaultSSLSocketFactory . . . . .	1663
decaf::internal::net::ssl::openssl::OpenSSLSocketFactory . . . . .	2825
decaf::net::SocketImplFactory . . . . .	3481
decaf::net::SocketOptions . . . . .	3482
decaf::net::SocketImpl . . . . .	3472
decaf::internal::net::tcp::TcpSocket . . . . .	3681
decaf::net::ssl::SSLContext . . . . .	3489
decaf::net::ssl::SSLContextSpi . . . . .	3492
decaf::internal::net::ssl::openssl::OpenSSLContextSpi . . . . .	2792
decaf::net::ssl::SSLParameters . . . . .	3495
activemq::commands::BrokerError::StackTraceElement . . . . .	3521
cms::Startable . . . . .	3527
cms::Connection . . . . .	1232
decaf::lang::STATIC_CAST_TOKEN . . . . .	3528
activemq::core::ActiveMQConstants::StaticInitializer . . . . .	3528
activemq::wireformat::stomp::StompCommandConstants . . . . .	3571

activemq:wireformat:stomp:StompFrame	3576
activemq:wireformat:stomp:StompHelper	3581
cms:Stoppable	3590
cms:Connection	1232
decaf:util:StringTokenizer	3613
decaf:util:concurrent:Synchronizable	3644
activemq:core:MessageDispatchChannel	2559
decaf:util:Collection< PrimitiveValueNode >	1155
decaf:internal:util:concurrent:SynchronizableImpl	3655
decaf:io:InputStream	2002
decaf:io:OutputStream	2856
decaf:util:Collection< E >	1155
decaf:util:concurrent:Mutex	2736
decaf:util:Map< K, V, COMPARATOR >	2419
decaf:util:AbstractMap< K, V, COMPARATOR >	162
decaf:util:concurrent:ConcurrentMap< K, V, COMPARATOR >	1198
decaf:util:concurrent:ConcurrentStlMap< K, V, COMPARATOR >	1203
decaf:util:StlMap< K, V, COMPARATOR >	3543
decaf:util:StlQueue< T >	3556
decaf:util:Map< std::string, PrimitiveValueNode, std::less< std::string > >	2419
decaf:util:StlMap< std::string, PrimitiveValueNode >	3543
activemq:util:PrimitiveMap	2941
activemq:core:Synchronization	3659
decaf:lang:System	3670
activemq:threads:Task	3678
activemq:core:ActiveMQSessionExecutor	503
activemq:threads:CompositeTask	1193
activemq:transport:failover:BackupTransportPool	720
activemq:transport:failover:CloseTransportsTask	1122
activemq:transport:failover:FailoverTransport	1835
activemq:threads:CompositeTaskRunner	1194
decaf:util:concurrent:TaskListener	3679
activemq:threads:TaskRunner	3680
activemq:threads:CompositeTaskRunner	1194
activemq:threads:DedicatedTaskRunner	1638
decaf:util:concurrent:ThreadFactory	3716
decaf:lang:ThreadGroup	3717
decaf:lang:Throwable	3724
decaf:lang:Exception	1794
activemq:exceptions:ActiveMQException	328
activemq:exceptions:BrokerException	827
decaf:io:IOException	2103
decaf:io:EOFException	1789
decaf:io:InterruptedIOException	2089
decaf:net:SocketTimeoutException	3487
decaf:io:UnsupportedEncodingException	3847
decaf:io:UTFDataFormatException	3897

decaf::net::HttpRetryException . . . . .	1948
decaf::net::MalformedURLException . . . . .	2416
decaf::net::ProtocolException . . . . .	3083
decaf::net::SocketException . . . . .	3465
decaf::internal::net::ssl::openssl::OpenSSLSocketException . . . . .	2821
decaf::net::BindException . . . . .	797
decaf::net::ConnectException . . . . .	1230
decaf::net::NoRouteToHostException . . . . .	2773
decaf::net::PortUnreachableException . . . . .	2922
decaf::net::UnknownHostException . . . . .	3841
decaf::net::UnknownServiceException . . . . .	3844
decaf::util::zip::ZipException . . . . .	3991
decaf::lang::exceptions::ClassCastException . . . . .	1117
decaf::lang::exceptions::IllegalArgumentException . . . . .	1953
decaf::lang::exceptions::IllegalMonitorStateException . . . . .	1955
decaf::lang::exceptions::IllegalStateException . . . . .	1959
decaf::nio::InvalidMarkException . . . . .	2096
decaf::lang::exceptions::IllegalThreadStateException . . . . .	1962
decaf::lang::exceptions::IndexOutOfBoundsException . . . . .	1967
decaf::lang::exceptions::InterruptedException . . . . .	2086
decaf::lang::exceptions::InvalidStateException . . . . .	2100
decaf::lang::exceptions::NoSuchElementException . . . . .	2778
decaf::lang::exceptions::NullPointerException . . . . .	2783
decaf::lang::exceptions::NumberFormatException . . . . .	2789
decaf::lang::exceptions::RuntimeException . . . . .	3267
decaf::lang::exceptions::UnsupportedOperationException . . . . .	3849
decaf::nio::ReadOnlyBufferException . . . . .	3115
decaf::net::URISyntaxException . . . . .	3880
decaf::nio::BufferOverflowException . . . . .	914
decaf::nio::BufferUnderflowException . . . . .	916
decaf::security::GeneralSecurityException . . . . .	1934
decaf::security::cert::CertificateException . . . . .	1061
decaf::security::cert::CertificateEncodingException . . . . .	1059
decaf::security::cert::CertificateExpiredException . . . . .	1063
decaf::security::cert::CertificateNotYetValidException . . . . .	1065
decaf::security::cert::CertificateParsingException . . . . .	1067
decaf::security::KeyException . . . . .	2255
decaf::security::InvalidKeyException . . . . .	2094
decaf::security::KeyManagementException . . . . .	2257
decaf::security::NoSuchAlgorithmException . . . . .	2776
decaf::security::NoSuchProviderException . . . . .	2781
decaf::security::SignatureException . . . . .	3440
decaf::util::concurrent::BrokenBarrierException . . . . .	820
decaf::util::concurrent::CancellationException . . . . .	1052
decaf::util::concurrent::ExecutionException . . . . .	1829
decaf::util::concurrent::RejectedExecutionException . . . . .	3134
decaf::util::concurrent::TimeoutException . . . . .	3728
decaf::util::zip::DataFormatException . . . . .	1520
decaf::util::Timer . . . . .	3730

decaf::internal::util::TimerTaskHeap . . . . .	3745
activemq::state::TransactionState . . . . .	3813
decaf::internal::util::concurrent::Transferer< E > . . . . .	3815
decaf::internal::util::concurrent::TransferQueue< E > . . . . .	3815
decaf::internal::util::concurrent::TransferStack< E > . . . . .	3817
activemq::transport::TransportFactory . . . . .	3825
activemq::transport::AbstractTransportFactory . . . . .	170
activemq::transport::failover::FailoverTransportFactory . . . . .	1846
activemq::transport::mock::MockTransportFactory . . . . .	2734
activemq::transport::tcp::TcpTransportFactory . . . . .	3699
activemq::transport::tcp::SslTransportFactory . . . . .	3520
activemq::transport::TransportListener . . . . .	3836
activemq::core::ActiveMQConnection . . . . .	244
activemq::transport::DefaultTransportListener . . . . .	1670
activemq::transport::failover::BackupTransport . . . . .	718
activemq::transport::mock::InternalCommandListener . . . . .	2085
activemq::transport::failover::FailoverTransportListener . . . . .	1849
activemq::transport::TransportFilter . . . . .	3827
activemq::transport::TransportRegistry . . . . .	3837
tree_desc_s . . . . .	3840
decaf::lang::Thread::UncaughtExceptionHandler . . . . .	3841
decaf::internal::net::URIEncoderDecoder . . . . .	3865
decaf::internal::net::URIHelper . . . . .	3867
activemq::transport::failover::URIPool . . . . .	3875
activemq::util::URISupport . . . . .	3877
decaf::internal::net::URIType . . . . .	3884
decaf::net::URL . . . . .	3891
decaf::net::URLDecoder . . . . .	3893
decaf::net::URLEncoder . . . . .	3894
activemq::util::Usage . . . . .	3895
activemq::util::MemoryUsage . . . . .	2472
activemq::wireformat::WireFormat . . . . .	3907
activemq::wireformat::openwire::OpenWireFormat . . . . .	2837
activemq::wireformat::stomp::StompWireFormat . . . . .	3586
activemq::wireformat::WireFormatFactory . . . . .	3911
activemq::wireformat::openwire::OpenWireFormatFactory . . . . .	2849
activemq::wireformat::stomp::StompWireFormatFactory . . . . .	3589
activemq::wireformat::WireFormatRegistry . . . . .	3947
z_stream_s . . . . .	3990

## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<b>decaf::util::AbstractCollection</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>Collection</b> (p. 1155) interface, to minimize the effort required to implement this interface ) . . . . .	147
<b>decaf::util::AbstractList</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>List</b> (p. 2296) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array) ) . . . . .	161
<b>decaf::util::AbstractMap</b> < <b>K</b> , <b>V</b> , <b>COMPARATOR</b> > (This class provides a skeletal implementation of the <b>Map</b> (p. 2419) interface, to minimize the effort required to implement this interface ) . . . . .	162
<b>decaf::util::AbstractQueue</b> < <b>E</b> > (This class provides skeletal implementations of some <b>Queue</b> (p. 3094) operations ) . . . . .	163
<b>decaf::util::AbstractSequentialList</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>List</b> (p. 2296) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list) ) . . . . .	167
<b>decaf::util::AbstractSet</b> < <b>E</b> > (This class provides a skeletal implementation of the <b>Set</b> (p. 3379) interface to minimize the effort required to implement this interface ) . . . . .	168
<b>activemq::transport::AbstractTransportFactory</b> (Abstract implementation of the <b>TransportFactory</b> (p. 3825) interface, providing the base functionality that's common to most of the <b>TransportFactory</b> (p. 3825) instances ) . . . . .	170
<b>activemq::core::ActiveMQAckHandler</b> (Interface class that is used to give CMS Messages an interface to Ack themselves with ) . . . . .	171
<b>activemq::commands::ActiveMQBlobMessage</b> . . . . .	172
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessageMarshaller</b> (p. 177) ) . . . . .	177

<b>activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessage-</b> <b>Marshaller</b> (p. 182) ) . . . . .	182
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessage-</b> <b>Marshaller</b> (p. 186) ) . . . . .	186
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessage-</b> <b>Marshaller</b> (p. 190) ) . . . . .	190
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessage-</b> <b>Marshaller</b> (p. 194) ) . . . . .	194
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBlobMessage-</b> <b>Marshaller</b> (p. 198) ) . . . . .	198
<b>activemq::commands::ActiveMQBytesMessage</b> . . . . .	202
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMes-</b> <b>sageMarshaller</b> (p. 220) ) . . . . .	220
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMes-</b> <b>sageMarshaller</b> (p. 224) ) . . . . .	224
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMes-</b> <b>sageMarshaller</b> (p. 228) ) . . . . .	228
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMes-</b> <b>sageMarshaller</b> (p. 232) ) . . . . .	232
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMes-</b> <b>sageMarshaller</b> (p. 236) ) . . . . .	236
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQBytesMes-</b> <b>sageMarshaller</b> (p. 240) ) . . . . .	240
<b>activemq::core::ActiveMQConnection</b> (Concrete connection used for all con- nectors to the ActiveMQ broker ) . . . . .	244
<b>activemq::core::ActiveMQConnectionFactory</b> . . . . .	264
<b>activemq::core::ActiveMQConnectionMetaData</b> (This class houses all the various settings and information that is used by an instance of an <b>ActiveMQConnection</b> (p. 244) class ) . . . . .	275
<b>activemq::core::ActiveMQConstants</b> (Class holding constant values for var- ious ActiveMQ specific things Each constant is defined as an enu- meration and has functions that convert back an forth between string and enum values ) . . . . .	279
<b>activemq::core::ActiveMQConsumer</b> . . . . .	282
<b>activemq::library::ActiveMQCPP</b> . . . . .	292
<b>activemq::commands::ActiveMQDestination</b> . . . . .	293
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestination-</b> <b>Marshaller</b> (p. 304) ) . . . . .	304



<b>activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestination-</b> <b>Marshaller</b> (p. 308) ) . . . . .	308
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestination-</b> <b>Marshaller</b> (p. 312) ) . . . . .	312
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestination-</b> <b>Marshaller</b> (p. 316) ) . . . . .	316
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestination-</b> <b>Marshaller</b> (p. 320) ) . . . . .	320
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQDestination-</b> <b>Marshaller</b> (p. 324) ) . . . . .	324
<b>activemq::exceptions::ActiveMQException</b> . . . . .	328
<b>activemq::commands::ActiveMQMapMessage</b> . . . . .	330
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessage-</b> <b>Marshaller</b> (p. 344) ) . . . . .	344
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessage-</b> <b>Marshaller</b> (p. 348) ) . . . . .	348
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessage-</b> <b>Marshaller</b> (p. 352) ) . . . . .	352
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessage-</b> <b>Marshaller</b> (p. 356) ) . . . . .	356
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessage-</b> <b>Marshaller</b> (p. 360) ) . . . . .	360
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMapMessage-</b> <b>Marshaller</b> (p. 364) ) . . . . .	364
<b>activemq::commands::ActiveMQMessage</b> . . . . .	368
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMar-</b> <b>shaller</b> (p. 371) ) . . . . .	371
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMar-</b> <b>shaller</b> (p. 375) ) . . . . .	375
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMar-</b> <b>shaller</b> (p. 379) ) . . . . .	379
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMar-</b> <b>shaller</b> (p. 383) ) . . . . .	383

<b>activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 387) ) . . . . .	387
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQMessageMarshaller</b> (p. 391) ) . . . . .	391
<b>activemq::commands::ActiveMQMessageTemplate&lt; T &gt;</b> . . . . .	395
<b>activemq::commands::ActiveMQObjectMessage</b> . . . . .	414
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 416) ) . . . . .	416
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 421) ) . . . . .	421
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 425) ) . . . . .	425
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 429) ) . . . . .	429
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 433) ) . . . . .	433
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQObjectMessageMarshaller</b> (p. 437) ) . . . . .	437
<b>activemq::core::ActiveMQProducer</b> . . . . .	441
<b>activemq::util::ActiveMQProperties</b> (Implementation of the CMSProperties interface that delegates to a <b>decaf::util::Properties</b> (p. 3072) object ) . . . . .	449
<b>activemq::commands::ActiveMQQueue</b> . . . . .	453
<b>activemq::core::ActiveMQQueueBrowser</b> . . . . .	457
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 460) ) . . . . .	460
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 464) ) . . . . .	464
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 468) ) . . . . .	468
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 472) ) . . . . .	472
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 476) ) . . . . .	476
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQQueueMarshaller</b> (p. 480) ) . . . . .	480
<b>activemq::core::ActiveMQSession</b> . . . . .	484

<b>activemq::core::ActiveMQSessionExecutor</b> (Delegate dispatcher for a single session ) . . . . .	503
<b>activemq::commands::ActiveMQStreamMessage</b> . . . . .	506
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 523) ) . . . . .	523
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 527) ) . . . . .	527
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 531) ) . . . . .	531
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 535) ) . . . . .	535
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 539) ) . . . . .	539
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQStreamMessageMarshaller</b> (p. 543) ) . . . . .	543
<b>activemq::commands::ActiveMQTempDestination</b> . . . . .	547
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 551) ) . . . . .	551
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 555) ) . . . . .	555
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 558) ) . . . . .	558
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 562) ) . . . . .	562
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 566) ) . . . . .	566
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempDestinationMarshaller</b> (p. 570) ) . . . . .	570
<b>activemq::commands::ActiveMQTempQueue</b> . . . . .	574
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 578) ) . . . . .	578
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 582) ) . . . . .	582
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 586) ) . . . . .	586

<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 590) )	590
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 594) )	594
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempQueueMarshaller</b> (p. 598) )	598
<b>activemq::commands::ActiveMQTempTopic</b>	602
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 607) )	607
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 611) )	611
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 615) )	615
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 619) )	619
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 623) )	623
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTempTopicMarshaller</b> (p. 627) )	627
<b>activemq::commands::ActiveMQTextMessage</b>	631
<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 635) )	635
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 640) )	640
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 644) )	644
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 648) )	648
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 652) )	652
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTextMessageMarshaller</b> (p. 656) )	656
<b>activemq::commands::ActiveMQTopic</b>	660

<b>activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 664) ) . . . . .	664
<b>activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 668) ) . . . . .	668
<b>activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 672) ) . . . . .	672
<b>activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 676) ) . . . . .	676
<b>activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 680) ) . . . . .	680
<b>activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller</b> (Marshaling code for Open Wire Format for <b>ActiveMQTopicMarshaller</b> (p. 684) ) . . . . .	684
<b>activemq::core::ActiveMQTransactionContext</b> (Transaction Management class, hold messages that are to be redelivered upon a request to roll-back )	688
<b>decaf::zip::Adler32</b> (Class that can be used to compute an Adler-32 <b>Checksum</b> (p. 1114) for a data stream ) . . . . .	691
<b>decaf::lang::Appendable</b> (An object to which char sequences and values can be appended ) . . . . .	693
<b>decaf::internal::AprPool</b> (Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made ) . . . . .	696
<b>decaf::lang::ArrayPointer&lt; T, REFCOUNTER &gt;</b> (Decaf's implementation of a Smart <b>Pointer</b> (p. 2896) that is a template on a Type and is <b>Thread</b> (p. 3707) Safe if the default Reference Counter is used ) . . . . .	697
<b>decaf::lang::ArrayPointerComparator&lt; T, R &gt;</b> (This implementation of Comparator is designed to allow objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this <b>ArrayPointer</b> (p. 697) ) . . . . .	704
<b>decaf::util::concurrent::atomic::AtomicBoolean</b> (A boolean value that may be updated atomically ) . . . . .	705
<b>decaf::util::concurrent::atomic::AtomicInteger</b> (An int value that may be updated atomically ) . . . . .	708
<b>decaf::util::concurrent::atomic::AtomicRefCounter</b> . . . . .	713
<b>decaf::util::concurrent::atomic::AtomicReference&lt; T &gt;</b> (An Pointer reference that may be updated atomically ) . . . . .	716
<b>activemq::transport::failover::BackupTransport</b> . . . . .	718
<b>activemq::transport::failover::BackupTransportPool</b> . . . . .	720
<b>activemq::commands::BaseCommand</b> . . . . .	723
<b>activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 730) ) . . . . .	730
<b>activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 737) ) . . . . .	737

<b>activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 743) ) . . . . .	743
<b>activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 750) ) . . . . .	750
<b>activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 757) ) . . . . .	757
<b>activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>BaseCommandMarshaller</b> (p. 764) ) . . . . .	764
<b>activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller</b> (Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol ) . . . . .	770
<b>activemq::commands::BaseDataStructure</b> . . . . .	793
<b>binary_function</b> . . . . .	797
<b>decaf::net::BindException</b> . . . . .	797
<b>decaf::io::BlockingByteArrayInputStream</b> (This is a blocking version of a byte buffer stream ) . . . . .	800
<b>decaf::util::concurrent::BlockingQueue&lt; E &gt;</b> (A <b>decaf::util::Queue</b> (p. 3094) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element ) . . . . .	804
<b>decaf::lang::Boolean</b> . . . . .	810
<b>activemq::commands::BooleanExpression</b> . . . . .	816
<b>activemq::wireformat::openwire::utils::BooleanStream</b> (Manages the writing and reading of boolean data streams to and from a data source such as a <b>DataInputStream</b> or <b>DataOutputStream</b> ) . . . . .	818
<b>decaf::util::concurrent::BrokenBarrierException</b> . . . . .	820
<b>activemq::commands::BrokerError</b> (This class represents an Exception sent from the Broker ) . . . . .	823
<b>activemq::exceptions::BrokerException</b> . . . . .	827
<b>activemq::commands::BrokerId</b> . . . . .	828
<b>activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 832) ) . . . . .	832
<b>activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 836) ) . . . . .	836
<b>activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 840) ) . . . . .	840
<b>activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 844) ) . . . . .	844
<b>activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 848) ) . . . . .	848
<b>activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerIdMarshaller</b> (p. 852) ) . . . . .	852
<b>activemq::commands::BrokerInfo</b> . . . . .	856
<b>activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 862) ) . . . . .	862

<b>activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 867))	867
<b>activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 871))	871
<b>activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 875))	875
<b>activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 879))	879
<b>activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>BrokerInfoMarshaller</b> (p. 883))	883
<b>decaf::nio::Buffer</b> (A container for data of a specific primitive type)	887
<b>decaf::io::BufferedInputStream</b> (A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream)	893
<b>decaf::io::BufferedOutputStream</b> (Wrapper around another output stream that buffers output before writing to the target output stream)	899
<b>decaf::internal::nio::BufferFactory</b> (Factory class used by static methods in the <b>decaf::nio</b> (p. 136) package to create the various default version of the NIO interfaces)	901
<b>decaf::nio::BufferOverflowException</b>	914
<b>decaf::nio::BufferUnderflowException</b>	916
<b>decaf::lang::Byte</b>	918
<b>decaf::internal::util::ByteArrayAdapter</b> (This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data)	928
<b>decaf::internal::nio::ByteArrayBuffer</b> (This class defines six categories of operations upon byte buffers:)	951
<b>decaf::io::ByteArrayInputStream</b> (A <b>ByteArrayInputStream</b> (p. 984) contains an internal buffer that contains bytes that may be read from the stream)	984
<b>decaf::io::ByteArrayOutputStream</b>	992
<b>decaf::nio::ByteBuffer</b> (This class defines six categories of operations upon byte buffers:)	995
<b>cms::BytesMessage</b> (A <b>BytesMessage</b> (p. 1023) object is used to send a message containing a stream of unsigned bytes)	1023
<b>activemq::cmsutil::CachedConsumer</b> (A cached message consumer contained within a pooled session)	1041
<b>activemq::cmsutil::CachedProducer</b> (A cached message producer contained within a pooled session)	1044
<b>decaf::util::concurrent::Callable&lt; V &gt;</b> (A task that returns a result and may throw an exception)	1051
<b>decaf::util::concurrent::CancellationException</b>	1052
<b>decaf::security::cert::Certificate</b> (Base interface for all identity certificates)	1055
<b>decaf::security::cert::CertificateEncodingException</b>	1059
<b>decaf::security::cert::CertificateException</b>	1061

<b>decaf::security::cert::CertificateExpiredException</b>	1063
<b>decaf::security::cert::CertificateNotYetValidException</b>	1065
<b>decaf::security::cert::CertificateParsingException</b>	1067
<b>decaf::lang::Character</b>	1069
<b>decaf::internal::nio::CharArrayBuffer</b>	1077
<b>decaf::nio::CharBuffer</b> (This class defines four categories of operations upon character buffers: )	1089
<b>decaf::lang::CharSequence</b> (A <b>CharSequence</b> (p. 1107) is a readable sequence of char values )	1107
<b>decaf::util::zip::CheckedInputStream</b> (An implementation of a <b>FilterInputStream</b> that will maintain a <b>Checksum</b> (p. 1114) of the bytes read, the <b>Checksum</b> (p. 1114) can then be used to verify the integrity of the input stream )	1109
<b>decaf::util::zip::CheckedOutputStream</b> (An implementation of a <b>FilterOutputStream</b> that will maintain a <b>Checksum</b> (p. 1114) of the bytes written, the <b>Checksum</b> (p. 1114) can then be used to verify the integrity of the output stream )	1112
<b>decaf::util::zip::Checksum</b> (An interface used to represent <b>Checksum</b> (p. 1114) values in the Zip package )	1114
<b>decaf::lang::exceptions::ClassCastException</b>	1117
<b>cms::Closeable</b> (Interface for a class that implements the close method )	1119
<b>decaf::io::Closeable</b> (Interface for a class that implements the close method )	1120
<b>activemq::transport::failover::CloseTransportsTask</b>	1122
<b>activemq::cmsutil::CmsAccessor</b> (Base class for <b>activemq.cmsutil.CmsTemplate</b> (p. 1140) and other CMS-accessing gateway helpers, defining common properties such as the CMS <b>cms.ConnectionFactory</b> (p. 1294) to operate on )	1123
<b>activemq::cmsutil::CmsDestinationAccessor</b> (Extends the <b>CmsAccessor</b> (p. 1123) to add support for resolving destination names )	1127
<b>cms::CMSException</b> (CMS API Exception that is the base for all exceptions thrown from CMS classes )	1130
<b>activemq::util::CMSExceptionSupport</b>	1134
<b>cms::CMSProperties</b> (Interface for a Java-like properties object )	1135
<b>cms::CMSSecurityException</b> (This exception must be thrown when a provider rejects a user name/password submitted by a client )	1139
<b>activemq::cmsutil::CmsTemplate</b> ( <b>CmsTemplate</b> (p. 1140) simplifies performing synchronous CMS operations )	1140
<b>code</b>	1154
<b>decaf::util::Collection&lt; E &gt;</b> (The root interface in the collection hierarchy )	1155
<b>activemq::commands::Command</b>	1165
<b>activemq::state::CommandVisitor</b> (Interface for an Object that can visit the various Command Objects that are sent from and to this client )	1171
<b>activemq::state::CommandVisitorAdapter</b> (Default Implementation of a <b>CommandVisitor</b> (p. 1171) that returns NULL for all calls )	1179
<b>decaf::lang::Comparable&lt; T &gt;</b> (This interface imposes a total ordering on the objects of each class that implements it )	1186
<b>decaf::util::Comparator&lt; T &gt;</b> (A comparison function, which imposes a total ordering on some collection of objects )	1189
<b>activemq::util::CompositeData</b> (Represents a Composite URI )	1191



<b>activemq::threads::CompositeTask</b> (Represents a single task that can be part of a set of Tasks that are contained in a <b>CompositeTaskRunner</b> (p. 1194) ) . . . . .	1193
<b>activemq::threads::CompositeTaskRunner</b> (A <b>Task</b> (p. 3678) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added ) . . . . .	1194
<b>activemq::transport::CompositeTransport</b> (A Composite <b>Transport</b> (p. 3819) is a <b>Transport</b> (p. 3819) implementation that is composed of several Transports ) . . . . .	1197
<b>decaf::util::concurrent::ConcurrentMap&lt; K, V, COMPARATOR &gt;</b> (Interface for a <b>Map</b> (p. 2419) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available <b>Map</b> (p. 2419) interface ) . . . . .	1198
<b>decaf::util::concurrent::ConcurrentStlMap&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map ) . . . . .	1203
<b>decaf::util::concurrent::locks::Condition</b> ( <b>Condition</b> (p. 1220) factors out the <b>Mutex</b> (p. 2736) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary <b>Lock</b> (p. 2336) implementations ) . . . . .	1220
<b>decaf::util::concurrent::ConditionHandle</b> . . . . .	1226
<b>decaf::internal::util::concurrent::ConditionImpl</b> . . . . .	1228
<b>decaf::net::ConnectException</b> . . . . .	1230
<b>cms::Connection</b> (The client's connection to its provider ) . . . . .	1232
<b>activemq::commands::ConnectionControl</b> . . . . .	1237
<b>activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1242) ) . . . . .	1242
<b>activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1246) ) . . . . .	1246
<b>activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1250) ) . . . . .	1250
<b>activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1254) ) . . . . .	1254
<b>activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1258) ) . . . . .	1258
<b>activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionControlMarshaller</b> (p. 1262) ) . . . . .	1262
<b>activemq::commands::ConnectionError</b> . . . . .	1266
<b>activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1270) ) . . . . .	1270

<b>activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1274) ) . . . . .	1274
<b>activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1278) ) . . . . .	1278
<b>activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1282) ) . . . . .	1282
<b>activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1286) ) . . . . .	1286
<b>activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionErrorMarshaller</b> (p. 1290) ) . . . . .	1290
<b>cms::ConnectionFactory</b> (Defines the interface for a factory that creates connection objects, the <b>Connection</b> (p. 1232) objects returned implement the CMS <b>Connection</b> (p. 1232) interface and hide the CMS Provider specific implementation details behind that interface ) . . .	1294
<b>activemq::commands::ConnectionId</b> . . . . .	1297
<b>activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1301) ) . . . . .	1301
<b>activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1305) ) . . . . .	1305
<b>activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1309) ) . . . . .	1309
<b>activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1313) ) . . . . .	1313
<b>activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1317) ) . . . . .	1317
<b>activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionIdMarshaller</b> (p. 1321) ) . . . . .	1321
<b>activemq::commands::ConnectionInfo</b> . . . . .	1324
<b>activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1330) ) . . . . .	1330
<b>activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1335) ) . . . . .	1335
<b>activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1339) ) . . . . .	1339

<b>activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1343) ) . . . . .	1343
<b>activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1347) ) . . . . .	1347
<b>activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConnectionInfoMarshaller</b> (p. 1351) ) . . . . .	1351
<b>cms::ConnectionMetaData</b> (A <b>ConnectionMetaData</b> (p. 1355) object provides information describing the <b>Connection</b> (p. 1232) object ) . . .	1355
<b>activemq::state::ConnectionState</b> . . . . .	1358
<b>activemq::state::ConnectionStateTracker</b> . . . . .	1361
<b>decaf::util::logging::ConsoleHandler</b> (This <b>Handler</b> (p. 1941) publishes log records to System.err ) . . . . .	1367
<b>activemq::commands::ConsumerControl</b> . . . . .	1369
<b>activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1373) ) . . . . .	1373
<b>activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1378) ) . . . . .	1378
<b>activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1382) ) . . . . .	1382
<b>activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1386) ) . . . . .	1386
<b>activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1390) ) . . . . .	1390
<b>activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerControlMarshaller</b> (p. 1394) ) . . . . .	1394
<b>activemq::commands::ConsumerId</b> . . . . .	1398
<b>activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1402) ) . . . . .	1402
<b>activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1406) ) . . . . .	1406
<b>activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1410) ) . . . . .	1410
<b>activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1414) ) . . . . .	1414
<b>activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1418) ) . . . . .	1418

<b>activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerIdMarshaller</b> (p. 1422) ) . . . . .	1422
<b>activemq::commands::ConsumerInfo</b> . . . . .	1426
<b>activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1434) ) . . . . .	1434
<b>activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1439) ) . . . . .	1439
<b>activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1443) ) . . . . .	1443
<b>activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1447) ) . . . . .	1447
<b>activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1451) ) . . . . .	1451
<b>activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ConsumerInfoMarshaller</b> (p. 1455) ) . . . . .	1455
<b>activemq::state::ConsumerState</b> . . . . .	1459
<b>activemq::commands::ControlCommand</b> . . . . .	1459
<b>activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1462) ) . . . . .	1462
<b>activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1467) ) . . . . .	1467
<b>activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1471) ) . . . . .	1471
<b>activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1475) ) . . . . .	1475
<b>activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1479) ) . . . . .	1479
<b>activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ControlCommandMarshaller</b> (p. 1483) ) . . . . .	1483
<b>decaf::util::concurrent::CountDownLatch</b> . . . . .	1487
<b>decaf::util::zip::CRC32</b> (Class that can be used to compute a CRC-32 checksum for a data stream ) . . . . .	1490
<b>ct_data_s</b> . . . . .	1492
<b>activemq::commands::DataArrayResponse</b> . . . . .	1493
<b>activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponseMarshaller</b> (p. 1496) ) . . . . .	1496

<b>activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponse-</b> <b>Marshaller</b> (p. 1500) ) . . . . .	1500
<b>activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponse-</b> <b>Marshaller</b> (p. 1504) ) . . . . .	1504
<b>activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponse-</b> <b>Marshaller</b> (p. 1508) ) . . . . .	1508
<b>activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponse-</b> <b>Marshaller</b> (p. 1512) ) . . . . .	1512
<b>activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>DataArrayResponse-</b> <b>Marshaller</b> (p. 1516) ) . . . . .	1516
<b>decaf::util::zip::DataFormatException</b> . . . . .	1520
<b>decaf::io::DataInput</b> (The <b>DataInput</b> (p. 1523) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types ) . . . . .	1523
<b>decaf::io::DataInputStream</b> (A data input stream lets an application read primitive Java data types from an underlying input stream in a machine- independent way ) . . . . .	1532
<b>decaf::io::DataOutput</b> (The <b>DataOutput</b> (p. 1541) interface provides for con- verting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream ) . . . . .	1541
<b>decaf::io::DataOutputStream</b> (A data output stream lets an application write primitive Java data types to an output stream in a portable way ) . . . . .	1546
<b>activemq::commands::DataResponse</b> . . . . .	1550
<b>activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller</b> (Mar- shaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1553) ) . . . . .	1553
<b>activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller</b> (Mar- shaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1557) ) . . . . .	1557
<b>activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller</b> (Mar- shaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1561) ) . . . . .	1561
<b>activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller</b> (Mar- shaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1565) ) . . . . .	1565
<b>activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller</b> (Mar- shaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1569) ) . . . . .	1569
<b>activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller</b> (Mar- shaling code for Open Wire Format for <b>DataResponseMarshaller</b> (p. 1573) ) . . . . .	1573
<b>activemq::wireformat::openwire::marshal::DataStreamMarshaller</b> (Base class for all classes that marshal commands for Openwire ) . . . . .	1577
<b>activemq::commands::DataStructure</b> . . . . .	1628
<b>decaf::util::Date</b> (Wrapper class around a time value in milliseconds ) . . . . .	1633

<b>decaf::internal::DecafRuntime</b> (Handles APR initialization and termination ) .	1637
<b>activemq::threads::DedicatedTaskRunner</b> . . . . .	1638
<b>activemq::core::policies::DefaultPrefetchPolicy</b> . . . . .	1640
<b>activemq::core::policies::DefaultRedeliveryPolicy</b> . . . . .	1644
<b>decaf::internal::net::DefaultServerSocketFactory</b> (Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options ) . . . . .	1648
<b>decaf::internal::net::DefaultSocketFactory</b> (SocketFactory implementation that is used to create Sockets ) . . . . .	1652
<b>decaf::internal::net::ssl::DefaultSSLContext</b> (Default SSLContext manager for the Decaf library ) . . . . .	1657
<b>decaf::internal::net::ssl::DefaultSSLServerSocketFactory</b> (Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds ) . . . . .	1658
<b>decaf::internal::net::ssl::DefaultSSLSocketFactory</b> (Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds ) . . . . .	1663
<b>activemq::transport::DefaultTransportListener</b> . . . . .	1670
<b>decaf::util::zip::Deflater</b> (This class compresses data using the <i>DEFLATE</i> algorithm (see <i>specification</i> ) ) . . . . .	1672
<b>decaf::util::zip::DeflaterOutputStream</b> (Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream ) . . . . .	1682
<b>decaf::util::concurrent::Delayed</b> (A mix-in style interface for marking objects that should be acted upon after a given delay ) . . . . .	1686
<b>cms::DeliveryMode</b> (This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages ) . .	1687
<b>cms::Destination</b> (A <b>Destination</b> (p. 1688) object encapsulates a provider-specific address ) . . . . .	1688
<b>activemq::commands::ActiveMQDestination::DestinationFilter</b> . . . . .	1691
<b>activemq::commands::DestinationInfo</b> . . . . .	1691
<b>activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1696) ) . . . . .	1696
<b>activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1700) ) . . . . .	1700
<b>activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1704) ) . . . . .	1704
<b>activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1708) ) . . . . .	1708
<b>activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1712) ) . . . . .	1712

<b>activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>DestinationInfoMarshaller</b> (p. 1716) ) . . . . .	1716
<b>activemq::cmsutil::DestinationResolver</b> (Resolves a CMS destination name to a <b>Destination</b> ) . . . . .	1720
<b>activemq::commands::DiscoveryEvent</b> . . . . .	1722
<b>activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1725) ) . . . . .	1725
<b>activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1729) ) . . . . .	1729
<b>activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1733) ) . . . . .	1733
<b>activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1737) ) . . . . .	1737
<b>activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1741) ) . . . . .	1741
<b>activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller</b> (Marshaling code for Open Wire Format for <b>DiscoveryEventMarshaller</b> (p. 1745) ) . . . . .	1745
<b>activemq::core::DispatchData</b> (Simple POJO that contains the information necessary to route a message to a specified consumer ) . . . . .	1749
<b>activemq::core::Dispatcher</b> (Interface for an object responsible for dispatching messages to consumers ) . . . . .	1750
<b>decaf::lang::Double</b> . . . . .	1751
<b>decaf::internal::nio::DoubleArrayBuffer</b> . . . . .	1762
<b>decaf::nio::DoubleBuffer</b> (This class defines four categories of operations upon double buffers: ) . . . . .	1773
<b>decaf::lang::DYNAMIC_CAST_TOKEN</b> . . . . .	1786
<b>activemq::cmsutil::DynamicDestinationResolver</b> (Resolves a CMS destination name to a <b>Destination</b> ) . . . . .	1786
<b>decaf::util::Map&lt; K, V, COMPARATOR &gt;::Entry</b> . . . . .	1788
<b>decaf::io::EOFException</b> . . . . .	1789
<b>decaf::util::logging::ErrorManager</b> ( <b>ErrorManager</b> (p. 1792) objects can be attached to <b>Handlers</b> to process any error that occur on a <b>Handler</b> (p. 1941) during Logging ) . . . . .	1792
<b>decaf::lang::Exception</b> . . . . .	1794
<b>cms::ExceptionListener</b> (If a CMS provider detects a serious problem, it notifies the client application through an <b>ExceptionListener</b> (p. 1801) that is registered with the <b>Connection</b> (p. 1232) ) . . . . .	1801
<b>activemq::commands::ExceptionResponse</b> . . . . .	1802
<b>activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1804) ) . . . . .	1804

<b>activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1809) ) . . . . .	1809
<b>activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1813) ) . . . . .	1813
<b>activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1817) ) . . . . .	1817
<b>activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1821) ) . . . . .	1821
<b>activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ExceptionResponseMarshaller</b> (p. 1825) ) . . . . .	1825
<b>decaf::util::concurrent::ExecutionException</b> . . . . .	1829
<b>decaf::util::concurrent::Executor</b> (An object that executes submitted <b>decaf.lang Runnable</b> (p. 3264) tasks ) . . . . .	1831
<b>decaf::util::concurrent::ExecutorService</b> (An <b>Executor</b> (p. 1831) that provides methods to manage termination and methods that can produce a <b>Future</b> (p. 1929) for tracking progress of one or more asynchronous tasks ) . . . . .	1833
<b>activemq::transport::failover::FailoverTransport</b> . . . . .	1835
<b>activemq::transport::failover::FailoverTransportFactory</b> (Creates an instance of a <b>FailoverTransport</b> (p. 1835) ) . . . . .	1846
<b>activemq::transport::failover::FailoverTransportListener</b> (Utility class used by the <b>Transport</b> (p. 3819) to perform the work of responding to events from the active <b>Transport</b> (p. 3819) ) . . . . .	1849
<b>decaf::io::FileDescriptor</b> (This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files ) . . . . .	1850
<b>decaf::util::logging::Filter</b> (A <b>Filter</b> (p. 1853) can be used to provide fine grain control over what is logged, beyond the control provided by log levels ) . . . . .	1853
<b>decaf::io::FilterInputStream</b> (A <b>FilterInputStream</b> (p. 1854) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality ) . . . . .	1854
<b>decaf::io::FilterOutputStream</b> (This class is the superclass of all classes that filter output streams ) . . . . .	1861
<b>decaf::lang::Float</b> . . . . .	1865
<b>decaf::internal::nio::FloatArrayBuffer</b> . . . . .	1876
<b>decaf::nio::FloatBuffer</b> (This class defines four categories of operations upon float buffers: ) . . . . .	1887
<b>decaf::io::Flushable</b> (A <b>Flushable</b> (p. 1899) is a destination of data that can be flushed ) . . . . .	1899
<b>activemq::commands::FlushCommand</b> . . . . .	1900
<b>activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1903) ) . . . . .	1903



<b>activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1907) ) . . . . .	1907
<b>activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1911) ) . . . . .	1911
<b>activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1915) ) . . . . .	1915
<b>activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1919) ) . . . . .	1919
<b>activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>FlushCommandMarshaller</b> (p. 1923) ) . . . . .	1923
<b>decaf::util::logging::Formatter</b> (A <b>Formatter</b> (p. 1927) provides support for formatting <b>LogRecords</b> ) . . . . .	1927
<b>decaf::util::concurrent::Future&lt; V &gt;</b> (A <b>Future</b> (p. 1929) represents the re- sult of an asynchronous computation ) . . . . .	1929
<b>activemq::transport::correlator::FutureResponse</b> (A container that holds a response object ) . . . . .	1932
<b>decaf::security::GeneralSecurityException</b> . . . . .	1934
<b>decaf::internal::util::GenericResource&lt; T &gt;</b> (A <b>Generic Resource</b> (p. 3223) wraps some type and will delete it when the <b>Resource</b> (p. 3223) itself is deleted ) . . . . .	1937
<b>gz_header_s</b> . . . . .	1938
<b>gz_state</b> . . . . .	1939
<b>decaf::util::logging::Handler</b> (A <b>Handler</b> (p. 1941) object takes log messages from a <b>Logger</b> (p. 2345) and exports them ) . . . . .	1941
<b>decaf::internal::util::HexStringParser</b> . . . . .	1945
<b>activemq::wireformat::openwire::utils::HexTable</b> (Maps hexadecimal strings to the value of an index into the table, i.e ) . . . . .	1947
<b>decaf::net::HttpRetryException</b> . . . . .	1948
<b>activemq::util::IdGenerator</b> . . . . .	1951
<b>decaf::lang::exceptions::IllegalArgumentException</b> . . . . .	1953
<b>decaf::lang::exceptions::IllegalMonitorStateException</b> . . . . .	1955
<b>cms::IllegalStateException</b> (This exception is thrown when a method is in- voked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation ) . . . . .	1958
<b>decaf::lang::exceptions::IllegalStateException</b> . . . . .	1959
<b>decaf::lang::exceptions::IllegalThreadStateException</b> . . . . .	1962
<b>activemq::transport::inactivity::InactivityMonitor</b> . . . . .	1964
<b>decaf::lang::exceptions::IndexOutOfBoundsException</b> . . . . .	1967
<b>decaf::net::Inet4Address</b> . . . . .	1970
<b>decaf::net::Inet6Address</b> . . . . .	1973
<b>decaf::net::InetAddress</b> (Represents an IP address ) . . . . .	1974
<b>decaf::net::InetSocketAddress</b> . . . . .	1982
<b>inflate_state</b> . . . . .	1982
<b>decaf::util::zip::Inflater</b> (This class uncompresses data that was compressed using the <i>DEFLATE</i> algorithm (see <i>specification</i> ) ) . . . . .	1985

<b>decaf::util::zip::InflaterInputStream</b> (A <b>FilterInputStream</b> that decompresses data read from the wrapped <b>InputStream</b> instance ) . . . . .	1994
<b>decaf::io::InputStream</b> (A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes ) . . . . .	2002
<b>decaf::io::InputStreamReader</b> (An <b>InputStreamReader</b> (p. 2013) is a bridge from byte streams to character streams ) . . . . .	2013
<b>decaf::internal::nio::IntArrayBuffer</b> . . . . .	2015
<b>decaf::nio::IntBuffer</b> (This class defines four categories of operations upon <b>int</b> buffers: ) . . . . .	2026
<b>decaf::lang::Integer</b> . . . . .	2038
<b>activemq::commands::IntegerResponse</b> . . . . .	2054
<b>activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>IntegerResponseMarshaller</b> (p. 2057) ) . . . . .	2057
<b>activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>IntegerResponseMarshaller</b> (p. 2061) ) . . . . .	2061
<b>activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>IntegerResponseMarshaller</b> (p. 2065) ) . . . . .	2065
<b>activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>IntegerResponseMarshaller</b> (p. 2069) ) . . . . .	2069
<b>activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>IntegerResponseMarshaller</b> (p. 2073) ) . . . . .	2073
<b>activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>IntegerResponseMarshaller</b> (p. 2077) ) . . . . .	2077
<b>internal_state</b> . . . . .	2081
<b>activemq::transport::mock::InternalCommandListener</b> (Listens for Commands sent from the <b>MockTransport</b> (p. 2724) ) . . . . .	2085
<b>decaf::lang::exceptions::InterruptedException</b> . . . . .	2086
<b>decaf::io::InterruptedException</b> . . . . .	2089
<b>cms::InvalidClientIdException</b> (This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider ) . . . . .	2091
<b>cms::InvalidDestinationException</b> (This exception must be thrown when a destination either is not understood by a provider or is no longer valid ) . . . . .	2093
<b>decaf::security::InvalidKeyException</b> . . . . .	2094
<b>decaf::nio::InvalidMarkException</b> . . . . .	2096
<b>cms::InvalidSelectorException</b> (This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax ) . . . . .	2099
<b>decaf::lang::exceptions::InvalidStateException</b> . . . . .	2100
<b>decaf::io::IOException</b> . . . . .	2103
<b>activemq::transport::IOTransport</b> (Implementation of the <b>Transport</b> (p. 3819) interface that performs marshaling of commands to IO streams ) . . . . .	2105
<b>decaf::lang::Iterable&lt; E &gt;</b> (Implementing this interface allows an object to be cast to an <b>Iterable</b> (p. 2112) type for generic collections API calls ) . . . . .	2112

<b>decaf::util::Iterator</b> < T > (Defines an object that can be used to iterate over the elements of a collection )	2114
<b>activemq::commands::JournalQueueAck</b>	2116
<b>activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2119) )	2119
<b>activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2123) )	2123
<b>activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2127) )	2127
<b>activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2131) )	2131
<b>activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2135) )	2135
<b>activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalQueueAckMarshaller</b> (p. 2139) )	2139
<b>activemq::commands::JournalTopicAck</b>	2143
<b>activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2148) )	2148
<b>activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2152) )	2152
<b>activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2156) )	2156
<b>activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2160) )	2160
<b>activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2164) )	2164
<b>activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTopicAckMarshaller</b> (p. 2168) )	2168
<b>activemq::commands::JournalTrace</b>	2171
<b>activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2174) )	2174
<b>activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2178) )	2178
<b>activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2182) )	2182

<b>activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2186) ) . . . . .	2186
<b>activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2190) ) . . . . .	2190
<b>activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTraceMarshaller</b> (p. 2194) ) . . . . .	2194
<b>activemq::commands::JournalTransaction</b> . . . . .	2198
<b>activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2201) ) . . . . .	2201
<b>activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2205) ) . . . . .	2205
<b>activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2209) ) . . . . .	2209
<b>activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2213) ) . . . . .	2213
<b>activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2217) ) . . . . .	2217
<b>activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller</b> (Marshaling code for Open Wire Format for <b>JournalTransactionMarshaller</b> (p. 2221) ) . . . . .	2221
<b>activemq::commands::KeepAliveInfo</b> . . . . .	2225
<b>activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2228) ) . . . . .	2228
<b>activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2233) ) . . . . .	2233
<b>activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2237) ) . . . . .	2237
<b>activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2241) ) . . . . .	2241
<b>activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2245) ) . . . . .	2245
<b>activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>KeepAliveInfoMarshaller</b> (p. 2249) ) . . . . .	2249
<b>decaf::security::Key</b> (The <b>Key</b> (p. 2253) interface is the top-level interface for all keys ) . . . . .	2253
<b>decaf::security::KeyException</b> . . . . .	2255

<b>decaf::security::KeyManagementException</b> . . . . .	2257
<b>activemq::commands::LastPartialCommand</b> . . . . .	2260
<b>activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2262) ) . . . . .	2262
<b>activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2267) ) . . . . .	2267
<b>activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2271) ) . . . . .	2271
<b>activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2275) ) . . . . .	2275
<b>activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2279) ) . . . . .	2279
<b>activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>LastPartialCommand-</b> <b>Marshaller</b> (p. 2283) ) . . . . .	2283
<b>decaf::util::comparators::Less&lt; E &gt;</b> (Simple <b>Less</b> (p. 2287) <b>Comparator</b> (p. 1189) that compares to elements to determine if the first is less than the second ) . . . . .	2287
<b>std::less&lt; decaf::lang::ArrayPointer&lt; T &gt; &gt;</b> (An override of the less func- tion object so that the Pointer objects can be stored in STL Maps, etc ) . . . . .	2289
<b>std::less&lt; decaf::lang::Pointer&lt; T &gt; &gt;</b> (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc ) . . . . .	2289
<b>decaf::util::logging::Level</b> (Defines a set of standard logging levels that can be used to control logging output ) . . . . .	2290
<b>decaf::util::List&lt; E &gt;</b> (An ordered collection (also known as a sequence) ) . . . . .	2296
<b>decaf::util::ListIterator&lt; E &gt;</b> (An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list ) . . . . .	2303
<b>activemq::commands::LocalTransactionId</b> . . . . .	2306
<b>activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2310) ) . . . . .	2310
<b>activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2314) ) . . . . .	2314
<b>activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2318) ) . . . . .	2318
<b>activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2322) ) . . . . .	2322
<b>activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionId-</b> <b>Marshaller</b> (p. 2326) ) . . . . .	2326

<b>activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>LocalTransactionIdMarshaller</b> (p. 2330) ) . . . . .	2330
<b>decaf::util::concurrent::Lock</b> (A wrapper class around a given synchronization mechanism that provides automatic release upon destruction ) . . . . .	2334
<b>decaf::util::concurrent::locks::Lock</b> ( <b>Lock</b> (p. 2336) implementations provide more extensive locking operations than can be obtained using synchronized statements ) . . . . .	2336
<b>decaf::util::concurrent::locks::LockSupport</b> (Basic thread blocking primitives for creating locks and other synchronization classes ) . . . . .	2341
<b>decaf::util::logging::Logger</b> (A <b>Logger</b> (p. 2345) object is used to log messages for a specific system or application component ) . . . . .	2345
<b>decaf::util::logging::LoggerHierarchy</b> . . . . .	2357
<b>activemq::io::LoggingInputStream</b> . . . . .	2358
<b>activemq::io::LoggingOutputStream</b> (OutputStream filter that just logs the data being written ) . . . . .	2359
<b>activemq::transport::logging::LoggingTransport</b> (A transport filter that logs commands as they are sent/received ) . . . . .	2360
<b>decaf::util::logging::LogManager</b> (There is a single global <b>LogManager</b> (p. 2363) object that is used to maintain a set of shared state about Loggers and log services ) . . . . .	2363
<b>decaf::util::logging::LogRecord</b> ( <b>LogRecord</b> (p. 2370) objects are used to pass logging requests between the logging framework and individual log Handlers ) . . . . .	2370
<b>decaf::util::logging::LogWriter</b> . . . . .	2375
<b>decaf::lang::Long</b> . . . . .	2377
<b>decaf::internal::nio::LongArrayBuffer</b> . . . . .	2392
<b>decaf::nio::LongBuffer</b> (This class defines four categories of operations upon long long buffers: ) . . . . .	2403
<b>activemq::util::LongSequenceGenerator</b> (This class is used to generate a sequence of long long values that are incremented each time a new value is requested ) . . . . .	2415
<b>decaf::net::MalformedURLException</b> . . . . .	2416
<b>decaf::util::Map&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map ) . . . . .	2419
<b>cms::MapMessage</b> (A <b>MapMessage</b> (p. 2431) object is used to send a set of name-value pairs ) . . . . .	2431
<b>decaf::util::logging::MarkBlockLogger</b> (Defines a class that can be used to mark the entry and exit from scoped blocks ) . . . . .	2443
<b>activemq::wireformat::MarshalAware</b> . . . . .	2444
<b>activemq::wireformat::openwire::marshal::v6::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . . . . .	2447
<b>activemq::wireformat::openwire::marshal::v3::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . . . . .	2447
<b>activemq::wireformat::openwire::marshal::v4::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . . . . .	2448
<b>activemq::wireformat::openwire::marshal::v5::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . . . . .	2449

<b>activemq::wireformat::openwire::marshal::v1::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . . . .	2450
<b>activemq::wireformat::openwire::marshal::v2::MarshallerFactory</b> (Used to create marshallers for a specific version of the wire protocol ) . . . .	2450
<b>activemq::util::MarshallingSupport</b> . . . . .	2451
<b>decaf::lang::Math</b> (The class <b>Math</b> (p. 2455) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions ) . . . . .	2455
<b>activemq::util::MemoryUsage</b> . . . . .	2472
<b>activemq::commands::Message</b> . . . . .	2475
<b>cms::Message</b> (Root of all messages ) . . . . .	2493
<b>activemq::commands::MessageAck</b> . . . . .	2521
<b>activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2526) ) . . . . .	2526
<b>activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2530) ) . . . . .	2530
<b>activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2534) ) . . . . .	2534
<b>activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2538) ) . . . . .	2538
<b>activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2542) ) . . . . .	2542
<b>activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageAckMarshaller</b> (p. 2546) ) . . . . .	2546
<b>cms::MessageConsumer</b> (A client uses a <b>MessageConsumer</b> (p. 2550) to received messages from a destination ) . . . . .	2550
<b>activemq::cmsutil::MessageCreator</b> (Creates the user-defined message to be sent by the <b>CmsTemplate</b> (p. 1140) ) . . . . .	2554
<b>activemq::commands::MessageDispatch</b> . . . . .	2555
<b>activemq::core::MessageDispatchChannel</b> . . . . .	2559
<b>activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2566) ) . . . . .	2566
<b>activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2570) ) . . . . .	2570
<b>activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2574) ) . . . . .	2574
<b>activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2578) ) . . . . .	2578

<b>activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2582) ) . . . . .	2582
<b>activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchMarshaller</b> (p. 2586) ) . . . . .	2586
<b>activemq::commands::MessageDispatchNotification</b> . . . . .	2590
<b>activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2595) ) . . . . .	2595
<b>activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2599) ) . . . . .	2599
<b>activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2603) ) . . . . .	2603
<b>activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2607) ) . . . . .	2607
<b>activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2611) ) . . . . .	2611
<b>activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageDispatchNotificationMarshaller</b> (p. 2616) ) . . . . .	2616
<b>cms::MessageEnumeration</b> (Defines an object that enumerates a collection of Messages ) . . . . .	2620
<b>cms::MessageEOFException</b> (This exception must be thrown when an unexpected end of stream has been reached when a <b>StreamMessage</b> (p. 3595) or <b>BytesMessage</b> (p. 1023) is being read ) . . . . .	2621
<b>cms::MessageFormatException</b> (This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type ) . . . . .	2622
<b>activemq::commands::MessageId</b> . . . . .	2623
<b>activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2628) ) . . . . .	2628
<b>activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2632) ) . . . . .	2632
<b>activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2636) ) . . . . .	2636
<b>activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2640) ) . . . . .	2640
<b>activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2644) ) . . . . .	2644



<b>activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageIdMarshaller</b> (p. 2648) ) . . . . .	2648
<b>cms::MessageListener</b> (A <b>MessageListener</b> (p. 2652) object is used to receive asynchronously delivered messages ) . . . . .	2652
<b>activemq::wireformat::openwire::marshal::v5::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2653) ) . . . . .	2653
<b>activemq::wireformat::openwire::marshal::v3::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2657) ) . . . . .	2657
<b>activemq::wireformat::openwire::marshal::v2::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2661) ) . . . . .	2661
<b>activemq::wireformat::openwire::marshal::v4::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2666) ) . . . . .	2666
<b>activemq::wireformat::openwire::marshal::v1::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2670) ) . . . . .	2670
<b>activemq::wireformat::openwire::marshal::v6::MessageMarshaller</b> (Marshaling code for Open Wire Format for <b>MessageMarshaller</b> (p. 2674) ) . . . . .	2674
<b>cms::MessageNotReadableException</b> (This exception must be thrown when a CMS client attempts to read a write-only message ) . . . . .	2679
<b>cms::MessageNotWriteableException</b> (This exception must be thrown when a CMS client attempts to write to a read-only message ) . . . . .	2680
<b>cms::MessageProducer</b> (A client uses a <b>MessageProducer</b> (p. 2681) object to send messages to a <b>Destination</b> (p. 1688) ) . . . . .	2681
<b>activemq::wireformat::openwire::utils::MessagePropertyInterceptor</b> (Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties ) . . . . .	2689
<b>activemq::commands::MessagePull</b> . . . . .	2695
<b>activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2700) ) . . . . .	2700
<b>activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2704) ) . . . . .	2704
<b>activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2708) ) . . . . .	2708
<b>activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2712) ) . . . . .	2712
<b>activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2716) ) . . . . .	2716
<b>activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller</b> (Marshaling code for Open Wire Format for <b>MessagePullMarshaller</b> (p. 2720) ) . . . . .	2720

<b>activemq::transport::mock::MockTransport</b> (The <b>MockTransport</b> (p. 2724) defines a base level <b>Transport</b> (p. 3819) class that is intended to be used in place of an a regular protocol <b>Transport</b> (p. 3819) such as TCP ) . . . . .	2724
<b>activemq::transport::mock::MockTransportFactory</b> (Manufactures <b>MockTransport</b> s, which are objects that read from input streams and write to output streams ) . . . . .	2734
<b>decaf::util::concurrent::Mutex</b> ( <b>Mutex</b> (p. 2736) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java ) . . . . .	2736
<b>decaf::util::concurrent::MutexHandle</b> . . . . .	2741
<b>decaf::internal::util::concurrent::MutexImpl</b> . . . . .	2742
<b>decaf::internal::net::Network</b> (Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API ) . . . . .	2744
<b>activemq::commands::NetworkBridgeFilter</b> . . . . .	2746
<b>activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2749) ) . . . . .	2749
<b>activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2753) ) . . . . .	2753
<b>activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2757) ) . . . . .	2757
<b>activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2761) ) . . . . .	2761
<b>activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2765) ) . . . . .	2765
<b>activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller</b> (Marshaling code for Open Wire Format for <b>NetworkBridgeFilterMarshaller</b> (p. 2769) ) . . . . .	2769
<b>decaf::net::NoRouteToHostException</b> . . . . .	2773
<b>decaf::security::NoSuchAlgorithmException</b> . . . . .	2776
<b>decaf::lang::exceptions::NoSuchElementException</b> . . . . .	2778
<b>decaf::security::NoSuchProviderException</b> . . . . .	2781
<b>decaf::lang::exceptions::NullPointerException</b> . . . . .	2783
<b>decaf::lang::Number</b> (The abstract class <b>Number</b> (p. 2786) is the superclass of classes <b>Byte</b> (p. 918), <b>Double</b> (p. 1751), <b>Float</b> (p. 1865), <b>Integer</b> (p. 2038), <b>Long</b> (p. 2377), and <b>Short</b> (p. 3380) ) . . . . .	2786
<b>decaf::lang::exceptions::NumberFormatException</b> . . . . .	2789
<b>cms::ObjectMessage</b> (Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object ) . . . . .	2791
<b>decaf::internal::net::ssl::openssl::OpenSSLContextSpi</b> (Provides an <b>SSLContext</b> that wraps the <b>OpenSSL</b> API ) . . . . .	2792
<b>decaf::internal::net::ssl::openssl::OpenSSLParameters</b> (Container class for parameters that are Common to <b>OpenSSL</b> socket classes ) . . . . .	2795

<b>decaf::internal::net::ssl::openssl::OpenSSLServerSocket</b> (SSLServerSocket based on OpenSSL library code ) . . . . .	2797
<b>decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory</b> (SSLServerSocketFactory that creates Server Sockets that use OpenSSL ) . . . .	2803
<b>decaf::internal::net::ssl::openssl::OpenSSLSocket</b> (Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API ) . . . . .	2808
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketException</b> (Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack ) . . . . .	2821
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketFactory</b> (Client Socket Factory that creates SSL based client sockets using the OpenSSL library ) . . . . .	2825
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream</b> (An output stream for reading data from an OpenSSL Socket instance ) . . . . .	2832
<b>decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream</b> (Output-Stream implementation used to write data to an <b>OpenSSLSocket</b> (p. 2808) instance ) . . . . .	2835
<b>activemq::wireformat::openwire::OpenWireFormat</b> . . . . .	2837
<b>activemq::wireformat::openwire::OpenWireFormatFactory</b> . . . . .	2849
<b>activemq::wireformat::openwire::OpenWireFormatNegotiator</b> . . . . .	2851
<b>activemq::wireformat::openwire::OpenWireResponseBuilder</b> . . . . .	2854
<b>decaf::io::OutputStream</b> (Base interface for any class that wants to represent an output stream of bytes ) . . . . .	2856
<b>decaf::io::OutputStreamWriter</b> (A class for turning a character stream into a byte stream ) . . . . .	2864
<b>activemq::commands::PartialCommand</b> . . . . .	2866
<b>activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2870) ) . . . . .	2870
<b>activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2874) ) . . . . .	2874
<b>activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2878) ) . . . . .	2878
<b>activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2883) ) . . . . .	2883
<b>activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2887) ) . . . . .	2887
<b>activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>PartialCommandMarshaller</b> (p. 2891) ) . . . . .	2891
<b>decaf::lang::Pointer&lt; T, REFCOUNTER &gt;</b> (Decaf's implementation of a Smart <b>Pointer</b> (p. 2896) that is a template on a Type and is <b>Thread</b> (p. 3707) Safe if the default Reference Counter is used ) . . . . .	2896

<b>decaf::lang::PointerComparator</b> < <b>T</b> , <b>R</b> > (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the <b>Pointer</b> (p. 2896) instance ) .	2903
<b>activemq::cmsutil::PooledSession</b> (A pooled session object that wraps around a delegate session ) . . . . .	2904
<b>decaf::util::concurrent::PooledThread</b> . . . . .	2918
<b>decaf::concurrent::PooledThreadListener</b> (Abstract Listener Interface for users of <b>ThreadPool</b> (p. 3718) ) . . . . .	2920
<b>decaf::net::PortUnreachableException</b> . . . . .	2922
<b>activemq::core::PrefetchPolicy</b> (Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP )	2924
<b>activemq::util::PrimitiveList</b> (List of primitives ) . . . . .	2929
<b>activemq::util::PrimitiveMap</b> (Map of named primitives ) . . . . .	2941
<b>activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller</b> (This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire ) . . . . .	2951
<b>activemq::util::PrimitiveValueNode::PrimitiveValue</b> (Define a union type comprised of the various types ) . . . . .	2957
<b>activemq::util::PrimitiveValueConverter</b> (Class controls the conversion of data contained in a <b>PrimitiveValueNode</b> (p. 2960) from one type to another ) . . . . .	2959
<b>activemq::util::PrimitiveValueNode</b> (Class that wraps around a single value of one of the many types ) . . . . .	2960
<b>decaf::security::Principal</b> (Base interface for a principal, which can represent an individual or organization ) . . . . .	2974
<b>decaf::util::PriorityQueue</b> < <b>E</b> > (An unbounded priority queue based on a binary heap algorithm ) . . . . .	2975
<b>activemq::commands::ProducerAck</b> . . . . .	2984
<b>activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 2988) ) . . . . .	2988
<b>activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 2992) ) . . . . .	2992
<b>activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 2996) ) . . . . .	2996
<b>activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3000) ) . . . . .	3000
<b>activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3004) ) . . . . .	3004
<b>activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerAckMarshaller</b> (p. 3008) ) . . . . .	3008
<b>activemq::cmsutil::ProducerCallback</b> (Callback for sending a message to a CMS destination ) . . . . .	3012

<b>activemq::cmsutil::CmsTemplate::ProducerExecutor</b>	3013
<b>activemq::commands::ProducerId</b>	3014
<b>activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3019))	3019
<b>activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3023))	3023
<b>activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3027))	3027
<b>activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3031))	3031
<b>activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3035))	3035
<b>activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerIdMarshaller</b> (p. 3039))	3039
<b>activemq::commands::ProducerInfo</b>	3043
<b>activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3047))	3047
<b>activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3052))	3052
<b>activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3056))	3056
<b>activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3060))	3060
<b>activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3064))	3064
<b>activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ProducerInfoMarshaller</b> (p. 3068))	3068
<b>activemq::state::ProducerState</b>	3072
<b>decaf::util::Properties</b> (Java-like properties class for mapping string names to string values)	3072
<b>decaf::util::logging::PropertiesChangeListener</b> (Defines the interface that classes can use to listen for change events on <b>Properties</b> (p. 3072))	3082
<b>decaf::net::ProtocolException</b>	3083
<b>decaf::security::PublicKey</b> (A public key)	3086
<b>decaf::io::PushbackInputStream</b> (A <b>PushbackInputStream</b> (p. 3086) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte)	3086
<b>cms::Queue</b> (An interface encapsulating a provider-specific queue name)	3093

<b>decaf::util::Queue</b> < <b>E</b> > (A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection ) . . . . .	3094
<b>cms::QueueBrowser</b> (This class implements in interface for browsing the messages in a <b>Queue</b> (p. 3093) without removing them ) . . . . .	3098
<b>decaf::util::Random</b> ( <b>Random</b> (p. 3100) Value Generator which is used to generate a stream of pseudorandom numbers ) . . . . .	3100
<b>decaf::lang::Readable</b> (A <b>Readable</b> (p. 3106) is a source of characters ) . . . . .	3106
<b>activemq::transport::inactivity::ReadChecker</b> (Runnable class that is used by the { } ) . . . . .	3107
<b>decaf::io::Reader</b> . . . . .	3108
<b>decaf::nio::ReadOnlyBufferException</b> . . . . .	3115
<b>decaf::util::concurrent::locks::ReadWriteLock</b> (A <b>ReadWriteLock</b> (p. 3117) maintains a pair of associated locks, one for read-only operations and one for writing ) . . . . .	3117
<b>activemq::cmsutil::CmsTemplate::ReceiveExecutor</b> . . . . .	3119
<b>activemq::core::RedeliveryPolicy</b> (Interface for a <b>RedeliveryPolicy</b> (p. 3121) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back ) . . . . .	3121
<b>decaf::util::concurrent::locks::ReentrantLock</b> (A reentrant mutual exclusion <b>Lock</b> (p. 2336) with extended capabilities ) . . . . .	3126
<b>decaf::util::concurrent::RejectedExecutionException</b> . . . . .	3134
<b>decaf::util::concurrent::RejectedExecutionHandler</b> (A handler for tasks that cannot be executed by a <b>ThreadPoolExecutor</b> (p. ??) ) . . . . .	3136
<b>activemq::commands::RemoveInfo</b> . . . . .	3137
<b>activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3141) ) . . . . .	3141
<b>activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3145) ) . . . . .	3145
<b>activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3149) ) . . . . .	3149
<b>activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3153) ) . . . . .	3153
<b>activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3157) ) . . . . .	3157
<b>activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveInfoMarshaller</b> (p. 3161) ) . . . . .	3161
<b>activemq::commands::RemoveSubscriptionInfo</b> . . . . .	3165
<b>activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3169) ) . . . . .	3169
<b>activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscriptionInfoMarshaller</b> (p. 3174) ) . . . . .	3174

<b>activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3178) ) . . . . .	3178
<b>activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3182) ) . . . . .	3182
<b>activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3186) ) . . . . .	3186
<b>activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>RemoveSubscription-</b> <b>InfoMarshaller</b> (p. 3190) ) . . . . .	3190
<b>activemq::commands::ReplayCommand</b> . . . . .	3194
<b>activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMar-</b> <b>shaller</b> (p. 3197) ) . . . . .	3197
<b>activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMar-</b> <b>shaller</b> (p. 3201) ) . . . . .	3201
<b>activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMar-</b> <b>shaller</b> (p. 3205) ) . . . . .	3205
<b>activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMar-</b> <b>shaller</b> (p. 3209) ) . . . . .	3209
<b>activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMar-</b> <b>shaller</b> (p. 3213) ) . . . . .	3213
<b>activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller</b> (Marshaling code for Open Wire Format for <b>ReplayCommandMar-</b> <b>shaller</b> (p. 3217) ) . . . . .	3217
<b>activemq::cmsutil::CmsTemplate::ResolveProducerExecutor</b> . . . . .	3221
<b>activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor</b> . . . . .	3222
<b>decaf::internal::util::Resource</b> (Interface for all Managed Resources in De- cafe, these objects are added to the Runtime System and are de- stroyed at shutdown ) . . . . .	3223
<b>decaf::internal::util::ResourceLifecycleManager</b> . . . . .	3224
<b>activemq::cmsutil::ResourceLifecycleManager</b> (Manages the lifecycle of a set of CMS resources ) . . . . .	3224
<b>activemq::commands::Response</b> . . . . .	3227
<b>activemq::transport::mock::ResponseBuilder</b> (Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol ) . . . . .	3231
<b>activemq::transport::correlator::ResponseCorrelator</b> (This type of trans- port filter is responsible for correlating asynchronous responses with requests ) . . . . .	3232
<b>activemq::wireformat::openwire::marshal::v4::ResponseMarshaller</b> (Mar- shaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3236) ) . . . . .	3236

<b>activemq::wireformat::openwire::marshal::v2::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3241))	3241
<b>activemq::wireformat::openwire::marshal::v5::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3246))	3246
<b>activemq::wireformat::openwire::marshal::v3::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3250))	3250
<b>activemq::wireformat::openwire::marshal::v1::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3255))	3255
<b>activemq::wireformat::openwire::marshal::v6::ResponseMarshaller</b> (Marshaling code for Open Wire Format for <b>ResponseMarshaller</b> (p. 3260))	3260
<b>decaf::lang::Runnable</b> (Interface for a runnable object - defines a task that can be run by a thread)	3264
<b>decaf::lang::Runtime</b>	3265
<b>decaf::lang::exceptions::RuntimeException</b>	3267
<b>decaf::security::SecureRandom</b>	3269
<b>decaf::internal::security::SecureRandomImpl</b> (Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources)	3275
<b>decaf::security::SecureRandomSpi</b> (Interface class used by Security Service Providers to implement a source of secure random bytes)	3278
<b>decaf::util::concurrent::Semaphore</b> (A counting semaphore)	3280
<b>activemq::cmsutil::CmsTemplate::SendExecutor</b>	3290
<b>decaf::net::ServerSocket</b> (This class implements server sockets)	3292
<b>decaf::net::ServerSocketFactory</b> (Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies)	3301
<b>cms::Session</b> (A <b>Session</b> (p.3305) object is a single-threaded context for producing and consuming messages)	3305
<b>activemq::cmsutil::SessionCallback</b> (Callback for executing any number of operations on a provided CMS Session)	3319
<b>activemq::commands::SessionId</b>	3320
<b>activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3324))	3324
<b>activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3328))	3328
<b>activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3332))	3332
<b>activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3336))	3336



<b>activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3340))	3340
<b>activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionIdMarshaller</b> (p. 3344))	3344
<b>activemq::commands::SessionInfo</b>	3348
<b>activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3352))	3352
<b>activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3356))	3356
<b>activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3360))	3360
<b>activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3364))	3364
<b>activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3368))	3368
<b>activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SessionInfoMarshaller</b> (p. 3372))	3372
<b>activemq::cmsutil::SessionPool</b> (A pool of CMS sessions from the same connection and with the same acknowledge mode)	3376
<b>activemq::state::SessionState</b>	3378
<b>decaf::util::Set&lt; E &gt;</b> (A collection that contains no duplicate elements)	3379
<b>decaf::lang::Short</b>	3380
<b>decaf::internal::nio::ShortArrayBuffer</b>	3390
<b>decaf::nio::ShortBuffer</b> (This class defines four categories of operations upon short buffers: )	3401
<b>activemq::commands::ShutdownInfo</b>	3413
<b>activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3416))	3416
<b>activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3420))	3420
<b>activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3424))	3424
<b>activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3428))	3428
<b>activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3432))	3432

<b>activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>ShutdownInfoMarshaller</b> (p. 3436) ) . . . . .	3436
<b>decaf::security::SignatureException</b> . . . . .	3440
<b>decaf::util::logging::SimpleFormatter</b> (Print a brief summary of the <b>LogRecord</b> (p. 2370) in a human readable format ) . . . . .	3442
<b>decaf::util::logging::SimpleLogger</b> . . . . .	3444
<b>decaf::net::Socket</b> . . . . .	3445
<b>decaf::net::SocketAddress</b> (Base class for protocol specific <b>Socket</b> (p. 3445) addresses ) . . . . .	3463
<b>decaf::net::SocketError</b> (Static utility class to simplify handling of error codes for socket operations ) . . . . .	3464
<b>decaf::net::SocketException</b> (Exception for errors when manipulating sockets ) . . . . .	3465
<b>decaf::net::SocketFactory</b> (The <b>SocketFactory</b> (p. 3467) is used to create <b>Socket</b> (p. 3445) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations ) . . . . .	3467
<b>decaf::internal::net::SocketFileDescriptor</b> (File Descriptor type used internally by Decaf Socket objects ) . . . . .	3471
<b>decaf::net::SocketImpl</b> (Acts as a base class for all physical <b>Socket</b> (p. 3445) implementations ) . . . . .	3472
<b>decaf::net::SocketImplFactory</b> (Factory class interface for a Factory that creates SocketImpl objects ) . . . . .	3481
<b>decaf::net::SocketOptions</b> . . . . .	3482
<b>decaf::net::SocketTimeoutException</b> . . . . .	3487
<b>decaf::net::ssl::SSLContext</b> (Represents an implementation of the Secure <b>Socket</b> (p. 3445) Layer for streaming based sockets ) . . . . .	3489
<b>decaf::net::ssl::SSLContextSpi</b> (Defines the interface that should be provided by an <b>SSLContext</b> (p. 3489) provider ) . . . . .	3492
<b>decaf::net::ssl::SSLParameters</b> . . . . .	3495
<b>decaf::net::ssl::SSLServerSocket</b> (Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol ) . . . . .	3498
<b>decaf::net::ssl::SSLServerSocketFactory</b> (Factory class interface that provides methods to create SSL Server Sockets ) . . . . .	3504
<b>decaf::net::ssl::SSLSocket</b> . . . . .	3506
<b>decaf::net::ssl::SSLSocketFactory</b> (Factory class interface for a <b>SocketFactory</b> (p. 3467) that can create <b>SSLSocket</b> (p. 3506) objects ) . . . . .	3515
<b>activemq::transport::tcp::SslTransport</b> ( <b>Transport</b> (p. 3819) for connecting to a Broker using an SSL Socket ) . . . . .	3518
<b>activemq::transport::tcp::SslTransportFactory</b> . . . . .	3520
<b>activemq::commands::BrokerError::StackTraceElement</b> . . . . .	3521
<b>decaf::internal::io::StandardErrorOutputStream</b> (Wrapper Around the Standard error Output facility on the current platform ) . . . . .	3521
<b>decaf::internal::io::StandardInputStream</b> . . . . .	3524
<b>decaf::internal::io::StandardOutputStream</b> . . . . .	3525
<b>cms::Startable</b> (Interface for a class that implements the start method ) . . . . .	3527
<b>decaf::lang::STATIC_CAST_TOKEN</b> . . . . .	3528
<b>activemq::core::ActiveMQConstants::StaticInitializer</b> . . . . .	3528

<b>decaf::util::StlList&lt; E &gt;</b> ( <b>List</b> (p. 2296) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type ) . . . . .	3529
<b>decaf::util::StlMap&lt; K, V, COMPARATOR &gt;</b> ( <b>Map</b> (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map ) . . .	3543
<b>decaf::util::StlQueue&lt; T &gt;</b> ( <b>Queue</b> (p. 3094) class accepts messages with an psuh(m) command where m is the message to be queued ) .	3556
<b>decaf::util::StlSet&lt; E &gt;</b> ( <b>Set</b> (p. 3379) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set ) . . . . .	3564
<b>activemq::wireformat::stomp::StompCommandConstants</b> . . . . .	3571
<b>activemq::wireformat::stomp::StompFrame</b> (A Stomp-level message frame that encloses all messages to and from the broker ) . . . . .	3576
<b>activemq::wireformat::stomp::StompHelper</b> (Utility Methods used when marshaling to and from StompFrame's ) . . . . .	3581
<b>activemq::wireformat::stomp::StompWireFormat</b> . . . . .	3586
<b>activemq::wireformat::stomp::StompWireFormatFactory</b> (Factory used to create the Stomp Wire Format instance ) . . . . .	3589
<b>cms::Stoppable</b> (Interface for a class that implements the stop method ) . . .	3590
<b>decaf::util::logging::StreamHandler</b> (Stream based logging <b>Handler</b> (p. 1941) ) . . . . .	3591
<b>cms::StreamMessage</b> (Interface for a <b>StreamMessage</b> (p. 3595) ) . . . . .	3595
<b>decaf::lang::String</b> (Immutable sequence of chars ) . . . . .	3610
<b>decaf::util::StringTokenizer</b> . . . . .	3613
<b>activemq::commands::SubscriptionInfo</b> . . . . .	3616
<b>activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3620) ) . . . . .	3620
<b>activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3624) ) . . . . .	3624
<b>activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3628) ) . . . . .	3628
<b>activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3632) ) . . . . .	3632
<b>activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3636) ) . . . . .	3636
<b>activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>SubscriptionInfoMarshaller</b> (p. 3640) ) . . . . .	3640
<b>decaf::util::concurrent::Synchronizable</b> (The interface for all synchronizable objects (that is, objects that can be locked and unlocked) ) . . .	3644
<b>decaf::internal::util::concurrent::SynchronizableImpl</b> (A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance ) . . . . .	3655

<b>activemq::core::Synchronization</b> (Transacted Object <b>Synchronization</b> (p. 3659), used to sync the events of a Transaction with the items in the Transaction ) . . . . .	3659
<b>decaf::util::concurrent::SynchronousQueue&lt; E &gt;</b> (A <b>blocking queue</b> (p. 804) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa ) . . . . .	3660
<b>decaf::lang::System</b> (Static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays ) . . . . .	3670
<b>activemq::threads::Task</b> (Represents a unit of work that requires one or more iterations to complete ) . . . . .	3678
<b>decaf::util::concurrent::TaskListener</b> . . . . .	3679
<b>activemq::threads::TaskRunner</b> . . . . .	3680
<b>decaf::internal::net::tcp::TcpSocket</b> (Platform-independent implementation of the socket interface ) . . . . .	3681
<b>decaf::internal::net::tcp::TcpSocketInputStream</b> (Input stream for performing reads on a socket ) . . . . .	3691
<b>decaf::internal::net::tcp::TcpSocketOutputStream</b> (Output stream for performing write operations on a socket ) . . . . .	3694
<b>activemq::transport::tcp::TcpTransport</b> (Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an <b>IOTransport</b> (p. 2105) ) . . . . .	3696
<b>activemq::transport::tcp::TcpTransportFactory</b> (Factory Responsible for creating the <b>TcpTransport</b> (p. 3696) ) . . . . .	3699
<b>cms::TemporaryQueue</b> (Defines a Temporary <b>Queue</b> (p. 3093) based <b>Destination</b> (p. 1688) ) . . . . .	3701
<b>cms::TemporaryTopic</b> (Defines a Temporary <b>Topic</b> (p. 3757) based <b>Destination</b> (p. 1688) ) . . . . .	3703
<b>cms::TextMessage</b> (Interface for a text message ) . . . . .	3704
<b>decaf::lang::Thread</b> (A <b>Thread</b> (p. 3707) is a concurrent unit of execution ) . . . . .	3707
<b>decaf::util::concurrent::ThreadFactory</b> (Public interface <b>ThreadFactory</b> (p. 3716) ) . . . . .	3716
<b>decaf::lang::ThreadGroup</b> . . . . .	3717
<b>decaf::util::concurrent::ThreadPool</b> (Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks ) . . . . .	3718
<b>decaf::lang::Throwable</b> (This class represents an error that has occurred ) . . . . .	3724
<b>decaf::util::concurrent::TimeoutException</b> . . . . .	3728
<b>decaf::util::Timer</b> (A facility for threads to schedule tasks for future execution in a background thread ) . . . . .	3730
<b>decaf::util::TimerTask</b> (A Base class for a task object that can be scheduled for one-time or repeated execution by a <b>Timer</b> (p. 3730) ) . . . . .	3743
<b>decaf::internal::util::TimerTaskHeap</b> (A Binary Heap implemented specifically for the Timer class in Decaf Util ) . . . . .	3745
<b>decaf::util::concurrent::TimeUnit</b> (A <b>TimeUnit</b> (p. 3748) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units ) . . . . .	3748
<b>cms::Topic</b> (An interface encapsulating a provider-specific topic name ) . . . . .	3757
<b>activemq::state::Tracked</b> . . . . .	3758

<b>activemq::commands::TransactionId</b>	3759
<b>activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3763) )	3763
<b>activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3766) )	3766
<b>activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3770) )	3770
<b>activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3774) )	3774
<b>activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3778) )	3778
<b>activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionIdMarshaller</b> (p. 3781) )	3781
<b>activemq::commands::TransactionInfo</b>	3785
<b>activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3789) )	3789
<b>activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3793) )	3793
<b>activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3797) )	3797
<b>activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3801) )	3801
<b>activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3805) )	3805
<b>activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>TransactionInfoMarshaller</b> (p. 3809) )	3809
<b>activemq::state::TransactionState</b>	3813
<b>decaf::internal::util::concurrent::Transferer&lt; E &gt;</b> (Shared internal API for dual stacks and queues )	3815
<b>decaf::internal::util::concurrent::TransferQueue&lt; E &gt;</b> (This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers )	3815
<b>decaf::internal::util::concurrent::TransferStack&lt; E &gt;</b>	3817
<b>activemq::transport::Transport</b> (Interface for a transport layer for command objects )	3819
<b>activemq::transport::TransportFactory</b> (Defines the interface for Factories that create Transports or TransportFilters )	3825
<b>activemq::transport::TransportFilter</b> (A filter on the transport layer )	3827

<b>activemq::transport::TransportListener</b> (A listener of asynchronous exceptions from a command transport object ) . . . . .	3836
<b>activemq::transport::TransportRegistry</b> (Registry of all <b>Transport</b> (p. 3819) Factories that are available to the client at runtime ) . . . . .	3837
<b>tree_desc_s</b> . . . . .	3840
<b>decaf::lang::Thread::UncaughtExceptionHandler</b> (Interface for handlers invoked when a <b>Thread</b> (p.3707) abruptly terminates due to an uncaught exception ) . . . . .	3841
<b>decaf::net::UnknownHostException</b> . . . . .	3841
<b>decaf::net::UnknownServiceException</b> . . . . .	3844
<b>decaf::io::UnsupportedEncodingException</b> (Thrown when the the Character Encoding is not supported ) . . . . .	3847
<b>decaf::lang::exceptions::UnsupportedOperationException</b> . . . . .	3849
<b>cms::UnsupportedOperationException</b> (This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use ) . . . . .	3852
<b>decaf::net::URI</b> (This class represents an instance of a <b>URI</b> (p. 3853) as defined by RFC 2396 ) . . . . .	3853
<b>decaf::internal::net::URLEncoderDecoder</b> . . . . .	3865
<b>decaf::internal::net::URIHelper</b> (Helper class used by the URI classes in encoding and decoding of URI's ) . . . . .	3867
<b>activemq::transport::failover::URIPool</b> . . . . .	3875
<b>activemq::util::URISupport</b> . . . . .	3877
<b>decaf::net::URISyntaxException</b> . . . . .	3880
<b>decaf::internal::net::URIType</b> (Basic type object that holds data that composes a given URI ) . . . . .	3884
<b>decaf::net::URL</b> (Class <b>URL</b> (p. 3891) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web ) . . . . .	3891
<b>decaf::net::URLDecoder</b> . . . . .	3893
<b>decaf::net::URLEncoder</b> . . . . .	3894
<b>activemq::util::Usage</b> . . . . .	3895
<b>decaf::io::UTFDataFormatException</b> (Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered ) . . . . .	3897
<b>decaf::util::UUID</b> (A class that represents an immutable universally unique identifier ( <b>UUID</b> (p. 3900)) ) . . . . .	3900
<b>activemq::wireformat::WireFormat</b> (Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams ) . . . . .	3907
<b>activemq::wireformat::WireFormatFactory</b> (The <b>WireFormatFactory</b> (p. 3911) is the interface that all <b>WireFormatFactory</b> (p. 3911) classes must extend ) . . . . .	3911
<b>activemq::commands::WireFormatInfo</b> . . . . .	3912
<b>activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3923) ) . . . . .	3923
<b>activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3927) ) . . . . .	3927

<b>activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3931) ) . . . . .	3931
<b>activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3935) ) . . . . .	3935
<b>activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3939) ) . . . . .	3939
<b>activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller</b> (Marshaling code for Open Wire Format for <b>WireFormatInfoMarshaller</b> (p. 3943) ) . . . . .	3943
<b>activemq::wireformat::WireFormatNegotiator</b> (Defines a <b>WireFormatNegotiator</b> (p. 3946) which allows a <b>WireFormat</b> (p. 3907) to ) . . . . .	3946
<b>activemq::wireformat::WireFormatRegistry</b> (Registry of all <b>WireFormat</b> (p. 3907) Factories that are available to the client at runtime ) . . . . .	3947
<b>activemq::transport::inactivity::WriteChecker</b> (Runnable class used by the { } ) . . . . .	3950
<b>decaf::io::Writer</b> . . . . .	3951
<b>decaf::security::auth::x500::X500Principal</b> . . . . .	3957
<b>decaf::security::cert::X509Certificate</b> (Base interface for all identity certificates ) . . . . .	3958
<b>activemq::commands::XATransactionId</b> . . . . .	3960
<b>activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 3964) ) . . . . .	3964
<b>activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 3968) ) . . . . .	3968
<b>activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 3972) ) . . . . .	3972
<b>activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 3976) ) . . . . .	3976
<b>activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 3980) ) . . . . .	3980
<b>activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller</b> (Marshaling code for Open Wire Format for <b>XATransactionIdMarshaller</b> (p. 3984) ) . . . . .	3984
<b>decaf::util::logging::XMLFormatter</b> (Format a <b>LogRecord</b> (p. 2370) into a standard XML format ) . . . . .	3988
<b>z_stream_s</b> . . . . .	3990
<b>decaf::util::zip::ZipException</b> . . . . .	3991





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/main/activemq/cmsutil/ <b>CachedConsumer.h</b> . . . . .	3995
src/main/activemq/cmsutil/ <b>CachedProducer.h</b> . . . . .	3995
src/main/activemq/cmsutil/ <b>CmsAccessor.h</b> . . . . .	3996
src/main/activemq/cmsutil/ <b>CmsDestinationAccessor.h</b> . . . . .	3996
src/main/activemq/cmsutil/ <b>CmsTemplate.h</b> . . . . .	3997
src/main/activemq/cmsutil/ <b>DestinationResolver.h</b> . . . . .	3997
src/main/activemq/cmsutil/ <b>DynamicDestinationResolver.h</b> . . . . .	3998
src/main/activemq/cmsutil/ <b>MessageCreator.h</b> . . . . .	3998
src/main/activemq/cmsutil/ <b>PooledSession.h</b> . . . . .	3999
src/main/activemq/cmsutil/ <b>ProducerCallback.h</b> . . . . .	3999
src/main/activemq/cmsutil/ <b>ResourceLifecycleManager.h</b> . . . . .	4000
src/main/activemq/cmsutil/ <b>SessionCallback.h</b> . . . . .	4001
src/main/activemq/cmsutil/ <b>SessionPool.h</b> . . . . .	4001
src/main/activemq/commands/ <b>ActiveMQBlobMessage.h</b> . . . . .	4002
src/main/activemq/commands/ <b>ActiveMQBytesMessage.h</b> . . . . .	4002
src/main/activemq/commands/ <b>ActiveMQDestination.h</b> . . . . .	4003
src/main/activemq/commands/ <b>ActiveMQMapMessage.h</b> . . . . .	4004
src/main/activemq/commands/ <b>ActiveMQMessage.h</b> . . . . .	4004
src/main/activemq/commands/ <b>ActiveMQMessageTemplate.h</b> . . . . .	4005
src/main/activemq/commands/ <b>ActiveMQObjectMessage.h</b> . . . . .	4006
src/main/activemq/commands/ <b>ActiveMQQueue.h</b> . . . . .	4006
src/main/activemq/commands/ <b>ActiveMQStreamMessage.h</b> . . . . .	4007
src/main/activemq/commands/ <b>ActiveMQTempDestination.h</b> . . . . .	4007
src/main/activemq/commands/ <b>ActiveMQTempQueue.h</b> . . . . .	4008
src/main/activemq/commands/ <b>ActiveMQTempTopic.h</b> . . . . .	4009
src/main/activemq/commands/ <b>ActiveMQTextMessage.h</b> . . . . .	4009
src/main/activemq/commands/ <b>ActiveMQTopic.h</b> . . . . .	4010
src/main/activemq/commands/ <b>BaseCommand.h</b> . . . . .	4010
src/main/activemq/commands/ <b>BaseDataStructure.h</b> . . . . .	4011

src/main/activemq/commands/ <b>BooleanExpression.h</b>	4011
src/main/activemq/commands/ <b>BrokerError.h</b>	4012
src/main/activemq/commands/ <b>BrokerId.h</b>	4012
src/main/activemq/commands/ <b>BrokerInfo.h</b>	4013
src/main/activemq/commands/ <b>Command.h</b>	4013
src/main/activemq/commands/ <b>ConnectionControl.h</b>	4014
src/main/activemq/commands/ <b>ConnectionError.h</b>	4014
src/main/activemq/commands/ <b>ConnectionId.h</b>	4015
src/main/activemq/commands/ <b>ConnectionInfo.h</b>	4015
src/main/activemq/commands/ <b>ConsumerControl.h</b>	4016
src/main/activemq/commands/ <b>ConsumerId.h</b>	4016
src/main/activemq/commands/ <b>ConsumerInfo.h</b>	4017
src/main/activemq/commands/ <b>ControlCommand.h</b>	4018
src/main/activemq/commands/ <b>dataArrayResponse.h</b>	4018
src/main/activemq/commands/ <b>DataResponse.h</b>	4019
src/main/activemq/commands/ <b>DataStructure.h</b>	4019
src/main/activemq/commands/ <b>DestinationInfo.h</b>	4020
src/main/activemq/commands/ <b>DiscoveryEvent.h</b>	4020
src/main/activemq/commands/ <b>ExceptionResponse.h</b>	4021
src/main/activemq/commands/ <b>FlushCommand.h</b>	4021
src/main/activemq/commands/ <b>IntegerResponse.h</b>	4022
src/main/activemq/commands/ <b>JournalQueueAck.h</b>	4022
src/main/activemq/commands/ <b>JournalTopicAck.h</b>	4023
src/main/activemq/commands/ <b>JournalTrace.h</b>	4023
src/main/activemq/commands/ <b>JournalTransaction.h</b>	4024
src/main/activemq/commands/ <b>KeepAliveInfo.h</b>	4024
src/main/activemq/commands/ <b>LastPartialCommand.h</b>	4025
src/main/activemq/commands/ <b>LocalTransactionId.h</b>	4025
src/main/activemq/commands/ <b>Message.h</b>	4026
src/main/activemq/commands/ <b>MessageAck.h</b>	4027
src/main/activemq/commands/ <b>MessageDispatch.h</b>	4028
src/main/activemq/commands/ <b>MessageDispatchNotification.h</b>	4029
src/main/activemq/commands/ <b>MessageId.h</b>	4029
src/main/activemq/commands/ <b>MessagePull.h</b>	4030
src/main/activemq/commands/ <b>NetworkBridgeFilter.h</b>	4030
src/main/activemq/commands/ <b>PartialCommand.h</b>	4031
src/main/activemq/commands/ <b>ProducerAck.h</b>	4031
src/main/activemq/commands/ <b>ProducerId.h</b>	4032
src/main/activemq/commands/ <b>ProducerInfo.h</b>	4032
src/main/activemq/commands/ <b>RemoveInfo.h</b>	4033
src/main/activemq/commands/ <b>RemoveSubscriptionInfo.h</b>	4034
src/main/activemq/commands/ <b>ReplayCommand.h</b>	4034
src/main/activemq/commands/ <b>Response.h</b>	4035
src/main/activemq/commands/ <b>SessionId.h</b>	4035
src/main/activemq/commands/ <b>SessionInfo.h</b>	4036
src/main/activemq/commands/ <b>ShutdownInfo.h</b>	4036
src/main/activemq/commands/ <b>SubscriptionInfo.h</b>	4037
src/main/activemq/commands/ <b>TransactionId.h</b>	4037
src/main/activemq/commands/ <b>TransactionInfo.h</b>	4038
src/main/activemq/commands/ <b>WireFormatInfo.h</b>	4038

src/main/activemq/commands/XATransactionId.h . . . . .	4039
src/main/activemq/core/ActiveMQAckHandler.h . . . . .	4039
src/main/activemq/core/ActiveMQConnection.h . . . . .	4040
src/main/activemq/core/ActiveMQConnectionFactory.h . . . . .	4040
src/main/activemq/core/ActiveMQConnectionMetaData.h . . . . .	4041
src/main/activemq/core/ActiveMQConstants.h . . . . .	4041
src/main/activemq/core/ActiveMQConsumer.h . . . . .	4042
src/main/activemq/core/ActiveMQProducer.h . . . . .	4043
src/main/activemq/core/ActiveMQQueueBrowser.h . . . . .	4043
src/main/activemq/core/ActiveMQSession.h . . . . .	4044
src/main/activemq/core/ActiveMQSessionExecutor.h . . . . .	4045
src/main/activemq/core/ActiveMQTransactionContext.h . . . . .	4046
src/main/activemq/core/DispatchData.h . . . . .	4046
src/main/activemq/core/Dispatcher.h . . . . .	4047
src/main/activemq/core/MessageDispatchChannel.h . . . . .	4047
src/main/activemq/core/PrefetchPolicy.h . . . . .	4049
src/main/activemq/core/RedeliveryPolicy.h . . . . .	4049
src/main/activemq/core/Synchronization.h . . . . .	4050
src/main/activemq/core/policies/DefaultPrefetchPolicy.h . . . . .	4048
src/main/activemq/core/policies/DefaultRedeliveryPolicy.h . . . . .	4048
src/main/activemq/exceptions/ActiveMQException.h . . . . .	4050
src/main/activemq/exceptions/BrokerException.h . . . . .	4051
src/main/activemq/exceptions/ExceptionDefines.h . . . . .	4051
src/main/activemq/io/LoggingInputStream.h . . . . .	4055
src/main/activemq/io/LoggingOutputStream.h . . . . .	4056
src/main/activemq/library/ActiveMQCPP.h . . . . .	4056
src/main/activemq/state/CommandVisitor.h . . . . .	4057
src/main/activemq/state/CommandVisitorAdapter.h . . . . .	4057
src/main/activemq/state/ConnectionState.h . . . . .	4059
src/main/activemq/state/ConnectionStateTracker.h . . . . .	4059
src/main/activemq/state/ConsumerState.h . . . . .	4060
src/main/activemq/state/ProducerState.h . . . . .	4061
src/main/activemq/state/SessionState.h . . . . .	4061
src/main/activemq/state/Tracked.h . . . . .	4062
src/main/activemq/state/TransactionState.h . . . . .	4062
src/main/activemq/threads/CompositeTask.h . . . . .	4063
src/main/activemq/threads/CompositeTaskRunner.h . . . . .	4063
src/main/activemq/threads/DedicatedTaskRunner.h . . . . .	4064
src/main/activemq/threads/Task.h . . . . .	4065
src/main/activemq/threads/TaskRunner.h . . . . .	4065
src/main/activemq/transport/AbstractTransportFactory.h . . . . .	4065
src/main/activemq/transport/CompositeTransport.h . . . . .	4066
src/main/activemq/transport/DefaultTransportListener.h . . . . .	4068
src/main/activemq/transport/IOTransport.h . . . . .	4074
src/main/activemq/transport/Transport.h . . . . .	4081
src/main/activemq/transport/TransportFactory.h . . . . .	4081
src/main/activemq/transport/TransportFilter.h . . . . .	4082
src/main/activemq/transport/TransportListener.h . . . . .	4082
src/main/activemq/transport/TransportRegistry.h . . . . .	4083
src/main/activemq/transport/correlator/FutureResponse.h . . . . .	4067

src/main/activemq/transport/correlator/ <b>ResponseCorrelator.h</b> . . . . .	4067
src/main/activemq/transport/failover/ <b>BackupTransport.h</b> . . . . .	4068
src/main/activemq/transport/failover/ <b>BackupTransportPool.h</b> . . . . .	4069
src/main/activemq/transport/failover/ <b>CloseTransportsTask.h</b> . . . . .	4070
src/main/activemq/transport/failover/ <b>FailoverTransport.h</b> . . . . .	4070
src/main/activemq/transport/failover/ <b>FailoverTransportFactory.h</b> . . . . .	4071
src/main/activemq/transport/failover/ <b>FailoverTransportListener.h</b> . . . . .	4072
src/main/activemq/transport/failover/ <b>URIPool.h</b> . . . . .	4072
src/main/activemq/transport/inactivity/ <b>InactivityMonitor.h</b> . . . . .	4073
src/main/activemq/transport/inactivity/ <b>ReadChecker.h</b> . . . . .	4073
src/main/activemq/transport/inactivity/ <b>WriteChecker.h</b> . . . . .	4074
src/main/activemq/transport/logging/ <b>LoggingTransport.h</b> . . . . .	4075
src/main/activemq/transport/mock/ <b>InternalCommandListener.h</b> . . . . .	4076
src/main/activemq/transport/mock/ <b>MockTransport.h</b> . . . . .	4076
src/main/activemq/transport/mock/ <b>MockTransportFactory.h</b> . . . . .	4077
src/main/activemq/transport/mock/ <b>ResponseBuilder.h</b> . . . . .	4078
src/main/activemq/transport/tcp/ <b>SslTransport.h</b> . . . . .	4078
src/main/activemq/transport/tcp/ <b>SslTransportFactory.h</b> . . . . .	4079
src/main/activemq/transport/tcp/ <b>TcpTransport.h</b> . . . . .	4079
src/main/activemq/transport/tcp/ <b>TcpTransportFactory.h</b> . . . . .	4080
src/main/activemq/util/ <b>ActiveMQProperties.h</b> . . . . .	4083
src/main/activemq/util/ <b>CMSExceptionSupport.h</b> . . . . .	4084
src/main/activemq/util/ <b>CompositeData.h</b> . . . . .	4086
src/main/activemq/util/ <b>Config.h</b> . . . . .	4086
src/main/activemq/util/ <b>IdGenerator.h</b> . . . . .	4087
src/main/activemq/util/ <b>LongSequenceGenerator.h</b> . . . . .	4088
src/main/activemq/util/ <b>MarshallingSupport.h</b> . . . . .	4088
src/main/activemq/util/ <b>MemoryUsage.h</b> . . . . .	4089
src/main/activemq/util/ <b>PrimitiveList.h</b> . . . . .	4089
src/main/activemq/util/ <b>PrimitiveMap.h</b> . . . . .	4090
src/main/activemq/util/ <b>PrimitiveValueConverter.h</b> . . . . .	4090
src/main/activemq/util/ <b>PrimitiveValueNode.h</b> . . . . .	4091
src/main/activemq/util/ <b>URISupport.h</b> . . . . .	4091
src/main/activemq/util/ <b>Usage.h</b> . . . . .	4092
src/main/activemq/wireformat/ <b>MarshalAware.h</b> . . . . .	4092
src/main/activemq/wireformat/ <b>WireFormat.h</b> . . . . .	4369
src/main/activemq/wireformat/ <b>WireFormatFactory.h</b> . . . . .	4370
src/main/activemq/wireformat/ <b>WireFormatNegotiator.h</b> . . . . .	4370
src/main/activemq/wireformat/ <b>WireFormatRegistry.h</b> . . . . .	4371
src/main/activemq/wireformat/openwire/ <b>OpenWireFormat.h</b> . . . . .	4362
src/main/activemq/wireformat/openwire/ <b>OpenWireFormatFactory.h</b> . . . . .	4362
src/main/activemq/wireformat/openwire/ <b>OpenWireFormatNegotiator.h</b> . . . . .	4363
src/main/activemq/wireformat/openwire/ <b>OpenWireResponseBuilder.h</b> . . . . .	4364
src/main/activemq/wireformat/openwire/marshal/ <b>BaseDataStreamMarshaller.h</b>	
4093	
src/main/activemq/wireformat/openwire/marshal/ <b>DataStreamMarshaller.h</b> . . . . .	4093
src/main/activemq/wireformat/openwire/marshal/ <b>PrimitiveTypesMarshaller.h</b>	
4094	
src/main/activemq/wireformat/openwire/marshal/v1/ <b>ActiveMQBlobMessageMarshaller.h</b>	
4095	

src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQBytesMessageMarshaller.h**  
4099  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQDestinationMarshaller.h**  
4104  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQMapMessageMarshaller.h**  
4108  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQMessageMarshaller.h**  
4112  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQObjectMessageMarshaller.h**  
4117  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQQueueMarshaller.h**  
4121  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQStreamMessageMarshaller.h**  
4125  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempDestinationMarshaller.h**  
4130  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempQueueMarshaller.h**  
4134  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempTopicMarshaller.h**  
4139  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTextMessageMarshaller.h**  
4143  
src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTopicMarshaller.h**  
4147  
src/main/activemq/wireformat/openwire/marshal/v1/**BaseCommandMarshaller.h**  
4152  
src/main/activemq/wireformat/openwire/marshal/v1/**BrokerIdMarshaller.h** . . 4156  
src/main/activemq/wireformat/openwire/marshal/v1/**BrokerInfoMarshaller.h** . 4160  
src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionControlMarshaller.h**  
4164  
src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionErrorMarshaller.h**  
4169  
src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionIdMarshaller.h**  
4173  
src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionInfoMarshaller.h**  
4177  
src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerControlMarshaller.h**  
4182  
src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerIdMarshaller.h**  
4186  
src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerInfoMarshaller.h**  
4190  
src/main/activemq/wireformat/openwire/marshal/v1/**ControlCommandMarshaller.h**  
4195  
src/main/activemq/wireformat/openwire/marshal/v1/**DataArrayResponseMarshaller.h**  
4199  
src/main/activemq/wireformat/openwire/marshal/v1/**DataResponseMarshaller.h**  
4203  
src/main/activemq/wireformat/openwire/marshal/v1/**DestinationInfoMarshaller.h**  
4208

src/main/activemq/wireformat/openwire/marshal/v1/**DiscoveryEventMarshaller.h**  
 4212  
 src/main/activemq/wireformat/openwire/marshal/v1/**ExceptionResponseMarshaller.h**  
 4217  
 src/main/activemq/wireformat/openwire/marshal/v1/**FlushCommandMarshaller.h**  
 4221  
 src/main/activemq/wireformat/openwire/marshal/v1/**IntegerResponseMarshaller.h**  
 4225  
 src/main/activemq/wireformat/openwire/marshal/v1/**JournalQueueAckMarshaller.h**  
 4230  
 src/main/activemq/wireformat/openwire/marshal/v1/**JournalTopicAckMarshaller.h**  
 4234  
 src/main/activemq/wireformat/openwire/marshal/v1/**JournalTraceMarshaller.h**  
 4238  
 src/main/activemq/wireformat/openwire/marshal/v1/**JournalTransactionMarshaller.h**  
 4243  
 src/main/activemq/wireformat/openwire/marshal/v1/**KeepAliveInfoMarshaller.h**  
 4247  
 src/main/activemq/wireformat/openwire/marshal/v1/**LastPartialCommandMarshaller.h**  
 4251  
 src/main/activemq/wireformat/openwire/marshal/v1/**LocalTransactionIdMarshaller.h**  
 4256  
 src/main/activemq/wireformat/openwire/marshal/v1/**MarshallerFactory.h** . . . 4260  
 src/main/activemq/wireformat/openwire/marshal/v1/**MessageAckMarshaller.h**  
 4263  
 src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchMarshaller.h**  
 4267  
 src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchNotificationMarshaller.h**  
 4272  
 src/main/activemq/wireformat/openwire/marshal/v1/**MessageIdMarshaller.h** . 4276  
 src/main/activemq/wireformat/openwire/marshal/v1/**MessageMarshaller.h** . . 4281  
 src/main/activemq/wireformat/openwire/marshal/v1/**MessagePullMarshaller.h**  
 4285  
 src/main/activemq/wireformat/openwire/marshal/v1/**NetworkBridgeFilterMarshaller.h**  
 4289  
 src/main/activemq/wireformat/openwire/marshal/v1/**PartialCommandMarshaller.h**  
 4293  
 src/main/activemq/wireformat/openwire/marshal/v1/**ProducerAckMarshaller.h**  
 4298  
 src/main/activemq/wireformat/openwire/marshal/v1/**ProducerIdMarshaller.h** . 4302  
 src/main/activemq/wireformat/openwire/marshal/v1/**ProducerInfoMarshaller.h**  
 4306  
 src/main/activemq/wireformat/openwire/marshal/v1/**RemoveInfoMarshaller.h**  
 4311  
 src/main/activemq/wireformat/openwire/marshal/v1/**RemoveSubscriptionInfoMarshaller.h**  
 4315  
 src/main/activemq/wireformat/openwire/marshal/v1/**ReplayCommandMarshaller.h**  
 4319  
 src/main/activemq/wireformat/openwire/marshal/v1/**ResponseMarshaller.h** . 4323  
 src/main/activemq/wireformat/openwire/marshal/v1/**SessionIdMarshaller.h** . 4328

src/main/activemq/wireformat/openwire/marshal/v1/**SessionInfoMarshaller.h**  
4332  
src/main/activemq/wireformat/openwire/marshal/v1/**ShutdownInfoMarshaller.h**  
4336  
src/main/activemq/wireformat/openwire/marshal/v1/**SubscriptionInfoMarshaller.h**  
4340  
src/main/activemq/wireformat/openwire/marshal/v1/**TransactionIdMarshaller.h**  
4344  
src/main/activemq/wireformat/openwire/marshal/v1/**TransactionInfoMarshaller.h**  
4349  
src/main/activemq/wireformat/openwire/marshal/v1/**WireFormatInfoMarshaller.h**  
4353  
src/main/activemq/wireformat/openwire/marshal/v1/**XATransactionIdMarshaller.h**  
4357  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBlobMessageMarshaller.h**  
4096  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBytesMessageMarshaller.h**  
4100  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQDestinationMarshaller.h**  
4104  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMapMessageMarshaller.h**  
4109  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMessageMarshaller.h**  
4113  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQObjectMessageMarshaller.h**  
4117  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQQueueMarshaller.h**  
4122  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQStreamMessageMarshaller.h**  
4126  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempDestinationMarshaller.h**  
4130  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempQueueMarshaller.h**  
4135  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempTopicMarshaller.h**  
4139  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTextMessageMarshaller.h**  
4144  
src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTopicMarshaller.h**  
4148  
src/main/activemq/wireformat/openwire/marshal/v2/**BaseCommandMarshaller.h**  
4152  
src/main/activemq/wireformat/openwire/marshal/v2/**BrokerIdMarshaller.h** . . 4157  
src/main/activemq/wireformat/openwire/marshal/v2/**BrokerInfoMarshaller.h** . 4161  
src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionControlMarshaller.h**  
4165  
src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionErrorMarshaller.h**  
4169  
src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionIdMarshaller.h**  
4174

src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionInfoMarshaller.h**  
 4178  
 src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerControlMarshaller.h**  
 4182  
 src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerIdMarshaller.h**  
 4187  
 src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerInfoMarshaller.h**  
 4191  
 src/main/activemq/wireformat/openwire/marshal/v2/**ControlCommandMarshaller.h**  
 4195  
 src/main/activemq/wireformat/openwire/marshal/v2/**DataArrayResponseMarshaller.h**  
 4200  
 src/main/activemq/wireformat/openwire/marshal/v2/**DataResponseMarshaller.h**  
 4204  
 src/main/activemq/wireformat/openwire/marshal/v2/**DestinationInfoMarshaller.h**  
 4209  
 src/main/activemq/wireformat/openwire/marshal/v2/**DiscoveryEventMarshaller.h**  
 4213  
 src/main/activemq/wireformat/openwire/marshal/v2/**ExceptionResponseMarshaller.h**  
 4217  
 src/main/activemq/wireformat/openwire/marshal/v2/**FlushCommandMarshaller.h**  
 4222  
 src/main/activemq/wireformat/openwire/marshal/v2/**IntegerResponseMarshaller.h**  
 4226  
 src/main/activemq/wireformat/openwire/marshal/v2/**JournalQueueAckMarshaller.h**  
 4230  
 src/main/activemq/wireformat/openwire/marshal/v2/**JournalTopicAckMarshaller.h**  
 4235  
 src/main/activemq/wireformat/openwire/marshal/v2/**JournalTraceMarshaller.h**  
 4239  
 src/main/activemq/wireformat/openwire/marshal/v2/**JournalTransactionMarshaller.h**  
 4243  
 src/main/activemq/wireformat/openwire/marshal/v2/**KeepAliveInfoMarshaller.h**  
 4248  
 src/main/activemq/wireformat/openwire/marshal/v2/**LastPartialCommandMarshaller.h**  
 4252  
 src/main/activemq/wireformat/openwire/marshal/v2/**LocalTransactionIdMarshaller.h**  
 4257  
 src/main/activemq/wireformat/openwire/marshal/v2/**MarshallerFactory.h** . . . 4261  
 src/main/activemq/wireformat/openwire/marshal/v2/**MessageAckMarshaller.h**  
 4264  
 src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchMarshaller.h**  
 4268  
 src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchNotificationMarshaller.h**  
 4273  
 src/main/activemq/wireformat/openwire/marshal/v2/**MessageIdMarshaller.h** . 4277  
 src/main/activemq/wireformat/openwire/marshal/v2/**MessageMarshaller.h** . . 4281  
 src/main/activemq/wireformat/openwire/marshal/v2/**MessagePullMarshaller.h**  
 4285



src/main/activemq/wireformat/openwire/marshal/v2/**NetworkBridgeFilterMarshaller.h**  
4290

src/main/activemq/wireformat/openwire/marshal/v2/**PartialCommandMarshaller.h**  
4294

src/main/activemq/wireformat/openwire/marshal/v2/**ProducerAckMarshaller.h**  
4298

src/main/activemq/wireformat/openwire/marshal/v2/**ProducerIdMarshaller.h** . 4303

src/main/activemq/wireformat/openwire/marshal/v2/**ProducerInfoMarshaller.h**  
4307

src/main/activemq/wireformat/openwire/marshal/v2/**RemoveInfoMarshaller.h**  
4311

src/main/activemq/wireformat/openwire/marshal/v2/**RemoveSubscriptionInfoMarshaller.h**  
4315

src/main/activemq/wireformat/openwire/marshal/v2/**ReplayCommandMarshaller.h**  
4320

src/main/activemq/wireformat/openwire/marshal/v2/**ResponseMarshaller.h** . 4324

src/main/activemq/wireformat/openwire/marshal/v2/**SessionIdMarshaller.h** . 4328

src/main/activemq/wireformat/openwire/marshal/v2/**SessionInfoMarshaller.h**  
4332

src/main/activemq/wireformat/openwire/marshal/v2/**ShutdownInfoMarshaller.h**  
4336

src/main/activemq/wireformat/openwire/marshal/v2/**SubscriptionInfoMarshaller.h**  
4341

src/main/activemq/wireformat/openwire/marshal/v2/**TransactionIdMarshaller.h**  
4345

src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h**  
4349

src/main/activemq/wireformat/openwire/marshal/v2/**WireFormatInfoMarshaller.h**  
4354

src/main/activemq/wireformat/openwire/marshal/v2/**XATransactionIdMarshaller.h**  
4358

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBlobMessageMarshaller.h**  
4096

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBytesMessageMarshaller.h**  
4101

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQDestinationMarshaller.h**  
4105

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMapMessageMarshaller.h**  
4109

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMessageMarshaller.h**  
4114

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQObjectMessageMarshaller.h**  
4118

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQQueueMarshaller.h**  
4122

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQStreamMessageMarshaller.h**  
4127

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempDestinationMarshaller.h**  
4131

src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempQueueMarshaller.h**  
4136  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempTopicMarshaller.h**  
4140  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTextMessageMarshaller.h**  
4144  
src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTopicMarshaller.h**  
4149  
src/main/activemq/wireformat/openwire/marshal/v3/**BaseCommandMarshaller.h**  
4153  
src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h** . . 4157  
src/main/activemq/wireformat/openwire/marshal/v3/**BrokerInfoMarshaller.h** . 4162  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionControlMarshaller.h**  
4166  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionErrorMarshaller.h**  
4170  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionIdMarshaller.h**  
4174  
src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionInfoMarshaller.h**  
4179  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerControlMarshaller.h**  
4183  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerIdMarshaller.h**  
4187  
src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerInfoMarshaller.h**  
4192  
src/main/activemq/wireformat/openwire/marshal/v3/**ControlCommandMarshaller.h**  
4196  
src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**  
4201  
src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h**  
4205  
src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h**  
4209  
src/main/activemq/wireformat/openwire/marshal/v3/**DiscoveryEventMarshaller.h**  
4214  
src/main/activemq/wireformat/openwire/marshal/v3/**ExceptionResponseMarshaller.h**  
4218  
src/main/activemq/wireformat/openwire/marshal/v3/**FlushCommandMarshaller.h**  
4222  
src/main/activemq/wireformat/openwire/marshal/v3/**IntegerResponseMarshaller.h**  
4227  
src/main/activemq/wireformat/openwire/marshal/v3/**JournalQueueAckMarshaller.h**  
4231  
src/main/activemq/wireformat/openwire/marshal/v3/**JournalTopicAckMarshaller.h**  
4235  
src/main/activemq/wireformat/openwire/marshal/v3/**JournalTraceMarshaller.h**  
4240  
src/main/activemq/wireformat/openwire/marshal/v3/**JournalTransactionMarshaller.h**  
4244

src/main/activemq/wireformat/openwire/marshal/v3/**KeepAliveInfoMarshaller.h**  
4249

src/main/activemq/wireformat/openwire/marshal/v3/**LastPartialCommandMarshaller.h**  
4253

src/main/activemq/wireformat/openwire/marshal/v3/**LocalTransactionIdMarshaller.h**  
4257

src/main/activemq/wireformat/openwire/marshal/v3/**MarshallerFactory.h** . . . 4261

src/main/activemq/wireformat/openwire/marshal/v3/**MessageAckMarshaller.h**  
4265

src/main/activemq/wireformat/openwire/marshal/v3/**MessageDispatchMarshaller.h**  
4269

src/main/activemq/wireformat/openwire/marshal/v3/**MessageDispatchNotificationMarshaller.h**  
4273

src/main/activemq/wireformat/openwire/marshal/v3/**MessageIdMarshaller.h** . 4278

src/main/activemq/wireformat/openwire/marshal/v3/**MessageMarshaller.h** . . 4282

src/main/activemq/wireformat/openwire/marshal/v3/**MessagePullMarshaller.h**  
4286

src/main/activemq/wireformat/openwire/marshal/v3/**NetworkBridgeFilterMarshaller.h**  
4290

src/main/activemq/wireformat/openwire/marshal/v3/**PartialCommandMarshaller.h**  
4295

src/main/activemq/wireformat/openwire/marshal/v3/**ProducerAckMarshaller.h**  
4299

src/main/activemq/wireformat/openwire/marshal/v3/**ProducerIdMarshaller.h** . 4303

src/main/activemq/wireformat/openwire/marshal/v3/**ProducerInfoMarshaller.h**  
4308

src/main/activemq/wireformat/openwire/marshal/v3/**RemoveInfoMarshaller.h**  
4312

src/main/activemq/wireformat/openwire/marshal/v3/**RemoveSubscriptionInfoMarshaller.h**  
4316

src/main/activemq/wireformat/openwire/marshal/v3/**ReplayCommandMarshaller.h**  
4320

src/main/activemq/wireformat/openwire/marshal/v3/**ResponseMarshaller.h** . 4325

src/main/activemq/wireformat/openwire/marshal/v3/**SessionIdMarshaller.h** . 4329

src/main/activemq/wireformat/openwire/marshal/v3/**SessionInfoMarshaller.h**  
4333

src/main/activemq/wireformat/openwire/marshal/v3/**ShutdownInfoMarshaller.h**  
4337

src/main/activemq/wireformat/openwire/marshal/v3/**SubscriptionInfoMarshaller.h**  
4341

src/main/activemq/wireformat/openwire/marshal/v3/**TransactionIdMarshaller.h**  
4346

src/main/activemq/wireformat/openwire/marshal/v3/**TransactionInfoMarshaller.h**  
4350

src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**  
4354

src/main/activemq/wireformat/openwire/marshal/v3/**XATransactionIdMarshaller.h**  
4359

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQBlobMessageMarshaller.h**  
4097

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQBytesMessageMarshaller.h**  
4101

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQDestinationMarshaller.h**  
4106

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMapMessageMarshaller.h**  
4110

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMessageMarshaller.h**  
4114

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQObjectMessageMarshaller.h**  
4119

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQQueueMarshaller.h**  
4123

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQStreamMessageMarshaller.h**  
4128

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempDestinationMarshaller.h**  
4132

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempQueueMarshaller.h**  
4136

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempTopicMarshaller.h**  
4141

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTextMessageMarshaller.h**  
4145

src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTopicMarshaller.h**  
4149

src/main/activemq/wireformat/openwire/marshal/v4/**BaseCommandMarshaller.h**  
4154

src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h** . . 4158

src/main/activemq/wireformat/openwire/marshal/v4/**BrokerInfoMarshaller.h** . 4162

src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionControlMarshaller.h**  
4166

src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionErrorMarshaller.h**  
4171

src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionIdMarshaller.h**  
4175

src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionInfoMarshaller.h**  
4179

src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerControlMarshaller.h**  
4184

src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerIdMarshaller.h**  
4188

src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerInfoMarshaller.h**  
4193

src/main/activemq/wireformat/openwire/marshal/v4/**ControlCommandMarshaller.h**  
4197

src/main/activemq/wireformat/openwire/marshal/v4/**DataArrayResponseMarshaller.h**  
4201

src/main/activemq/wireformat/openwire/marshal/v4/**DataResponseMarshaller.h**  
4206

src/main/activemq/wireformat/openwire/marshal/v4/**DestinationInfoMarshaller.h**  
4210

src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h	
4214	
src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h	
4219	
src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	
4223	
src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
4227	
src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
4232	
src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
4236	
src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	
4241	
src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
4245	
src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	
4249	
src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
4254	
src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
4258	
src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	4262
src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h	
4265	
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h	
4270	
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h	
4274	
src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h	4278
src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h	4283
src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h	
4287	
src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h	
4291	
src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h	
4295	
src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h	
4300	
src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h	4304
src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h	
4308	
src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h	
4313	
src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h	
4317	
src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h	
4321	
src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h	4325
src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h	4330

src/main/activemq/wireformat/openwire/marshal/v4/**SessionInfoMarshaller.h**  
4334  
src/main/activemq/wireformat/openwire/marshal/v4/**ShutdownInfoMarshaller.h**  
4338  
src/main/activemq/wireformat/openwire/marshal/v4/**SubscriptionInfoMarshaller.h**  
4342  
src/main/activemq/wireformat/openwire/marshal/v4/**TransactionIdMarshaller.h**  
4346  
src/main/activemq/wireformat/openwire/marshal/v4/**TransactionInfoMarshaller.h**  
4351  
src/main/activemq/wireformat/openwire/marshal/v4/**WireFormatInfoMarshaller.h**  
4355  
src/main/activemq/wireformat/openwire/marshal/v4/**XATransactionIdMarshaller.h**  
4360  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQBlobMessageMarshaller.h**  
4098  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQBytesMessageMarshaller.h**  
4102  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQDestinationMarshaller.h**  
4106  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQMapMessageMarshaller.h**  
4111  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQMessageMarshaller.h**  
4115  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQObjectMessageMarshaller.h**  
4120  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQQueueMarshaller.h**  
4124  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQStreamMessageMarshaller.h**  
4128  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempDestinationMarshaller.h**  
4133  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempQueueMarshaller.h**  
4137  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempTopicMarshaller.h**  
4141  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTextMessageMarshaller.h**  
4146  
src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTopicMarshaller.h**  
4150  
src/main/activemq/wireformat/openwire/marshal/v5/**BaseCommandMarshaller.h**  
4155  
src/main/activemq/wireformat/openwire/marshal/v5/**BrokerIdMarshaller.h** . . 4159  
src/main/activemq/wireformat/openwire/marshal/v5/**BrokerInfoMarshaller.h** . 4163  
src/main/activemq/wireformat/openwire/marshal/v5/**ConnectionControlMarshaller.h**  
4167  
src/main/activemq/wireformat/openwire/marshal/v5/**ConnectionErrorMarshaller.h**  
4171  
src/main/activemq/wireformat/openwire/marshal/v5/**ConnectionIdMarshaller.h**  
4176

src/main/activemq/wireformat/openwire/marshall/v5/ConnectionInfoMarshaller.h	
4180	
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerControlMarshaller.h	
4185	
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerIdMarshaller.h	
4189	
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerInfoMarshaller.h	
4193	
src/main/activemq/wireformat/openwire/marshall/v5/ControlCommandMarshaller.h	
4198	
src/main/activemq/wireformat/openwire/marshall/v5/DataArrayResponseMarshaller.h	
4202	
src/main/activemq/wireformat/openwire/marshall/v5/DataResponseMarshaller.h	
4206	
src/main/activemq/wireformat/openwire/marshall/v5/DestinationInfoMarshaller.h	
4211	
src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEventMarshaller.h	
4215	
src/main/activemq/wireformat/openwire/marshall/v5/ExceptionResponseMarshaller.h	
4219	
src/main/activemq/wireformat/openwire/marshall/v5/FlushCommandMarshaller.h	
4224	
src/main/activemq/wireformat/openwire/marshall/v5/IntegerResponseMarshaller.h	
4228	
src/main/activemq/wireformat/openwire/marshall/v5/JournalQueueAckMarshaller.h	
4233	
src/main/activemq/wireformat/openwire/marshall/v5/JournalTopicAckMarshaller.h	
4237	
src/main/activemq/wireformat/openwire/marshall/v5/JournalTraceMarshaller.h	
4241	
src/main/activemq/wireformat/openwire/marshall/v5/JournalTransactionMarshaller.h	
4246	
src/main/activemq/wireformat/openwire/marshall/v5/KeepAliveInfoMarshaller.h	
4250	
src/main/activemq/wireformat/openwire/marshall/v5/LastPartialCommandMarshaller.h	
4254	
src/main/activemq/wireformat/openwire/marshall/v5/LocalTransactionIdMarshaller.h	
4259	
src/main/activemq/wireformat/openwire/marshall/v5/MarshallerFactory.h	4262
src/main/activemq/wireformat/openwire/marshall/v5/MessageAckMarshaller.h	
4266	
src/main/activemq/wireformat/openwire/marshall/v5/MessageDispatchMarshaller.h	
4270	
src/main/activemq/wireformat/openwire/marshall/v5/MessageDispatchNotificationMarshaller.h	
4275	
src/main/activemq/wireformat/openwire/marshall/v5/MessageldMarshaller.h	4279
src/main/activemq/wireformat/openwire/marshall/v5/MessageMarshaller.h	4283
src/main/activemq/wireformat/openwire/marshall/v5/MessagePullMarshaller.h	
4287	

src/main/activemq/wireformat/openwire/marshal/v5/**NetworkBridgeFilterMarshaller.h**  
4292

src/main/activemq/wireformat/openwire/marshal/v5/**PartialCommandMarshaller.h**  
4296

src/main/activemq/wireformat/openwire/marshal/v5/**ProducerAckMarshaller.h**  
4301

src/main/activemq/wireformat/openwire/marshal/v5/**ProducerIdMarshaller.h** . 4305

src/main/activemq/wireformat/openwire/marshal/v5/**ProducerInfoMarshaller.h**  
4309

src/main/activemq/wireformat/openwire/marshal/v5/**RemoveInfoMarshaller.h**  
4313

src/main/activemq/wireformat/openwire/marshal/v5/**RemoveSubscriptionInfoMarshaller.h**  
4317

src/main/activemq/wireformat/openwire/marshal/v5/**ReplayCommandMarshaller.h**  
4322

src/main/activemq/wireformat/openwire/marshal/v5/**ResponseMarshaller.h** . 4326

src/main/activemq/wireformat/openwire/marshal/v5/**SessionIdMarshaller.h** . 4330

src/main/activemq/wireformat/openwire/marshal/v5/**SessionInfoMarshaller.h**  
4334

src/main/activemq/wireformat/openwire/marshal/v5/**ShutdownInfoMarshaller.h**  
4338

src/main/activemq/wireformat/openwire/marshal/v5/**SubscriptionInfoMarshaller.h**  
4343

src/main/activemq/wireformat/openwire/marshal/v5/**TransactionIdMarshaller.h**  
4347

src/main/activemq/wireformat/openwire/marshal/v5/**TransactionInfoMarshaller.h**  
4352

src/main/activemq/wireformat/openwire/marshal/v5/**WireFormatInfoMarshaller.h**  
4356

src/main/activemq/wireformat/openwire/marshal/v5/**XATransactionIdMarshaller.h**  
4360

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQBlobMessageMarshaller.h**  
4098

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQBytesMessageMarshaller.h**  
4103

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQDestinationMarshaller.h**  
4107

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQMapMessageMarshaller.h**  
4112

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQMessageMarshaller.h**  
4116

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQObjectMessageMarshaller.h**  
4120

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQQueueMarshaller.h**  
4125

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQStreamMessageMarshaller.h**  
4129

src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempDestinationMarshaller.h**  
4133



src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempQueueMarshaller.h**  
4138  
src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempTopicMarshaller.h**  
4142  
src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTextMessageMarshaller.h**  
4147  
src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTopicMarshaller.h**  
4151  
src/main/activemq/wireformat/openwire/marshal/v6/**BaseCommandMarshaller.h**  
4155  
src/main/activemq/wireformat/openwire/marshal/v6/**BrokerIdMarshaller.h** . . 4160  
src/main/activemq/wireformat/openwire/marshal/v6/**BrokerInfoMarshaller.h** . 4164  
src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionControlMarshaller.h**  
4168  
src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionErrorMarshaller.h**  
4172  
src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionIdMarshaller.h**  
4177  
src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionInfoMarshaller.h**  
4181  
src/main/activemq/wireformat/openwire/marshal/v6/**ConsumerControlMarshaller.h**  
4185  
src/main/activemq/wireformat/openwire/marshal/v6/**ConsumerIdMarshaller.h**  
4190  
src/main/activemq/wireformat/openwire/marshal/v6/**ConsumerInfoMarshaller.h**  
4194  
src/main/activemq/wireformat/openwire/marshal/v6/**ControlCommandMarshaller.h**  
4198  
src/main/activemq/wireformat/openwire/marshal/v6/**DataArrayResponseMarshaller.h**  
4203  
src/main/activemq/wireformat/openwire/marshal/v6/**DataResponseMarshaller.h**  
4207  
src/main/activemq/wireformat/openwire/marshal/v6/**DestinationInfoMarshaller.h**  
4211  
src/main/activemq/wireformat/openwire/marshal/v6/**DiscoveryEventMarshaller.h**  
4216  
src/main/activemq/wireformat/openwire/marshal/v6/**ExceptionResponseMarshaller.h**  
4220  
src/main/activemq/wireformat/openwire/marshal/v6/**FlushCommandMarshaller.h**  
4225  
src/main/activemq/wireformat/openwire/marshal/v6/**IntegerResponseMarshaller.h**  
4229  
src/main/activemq/wireformat/openwire/marshal/v6/**JournalQueueAckMarshaller.h**  
4233  
src/main/activemq/wireformat/openwire/marshal/v6/**JournalTopicAckMarshaller.h**  
4238  
src/main/activemq/wireformat/openwire/marshal/v6/**JournalTraceMarshaller.h**  
4242  
src/main/activemq/wireformat/openwire/marshal/v6/**JournalTransactionMarshaller.h**  
4246

src/main/activemq/wireformat/openwire/marshal/v6/**KeepAliveInfoMarshaller.h**  
 4251  
 src/main/activemq/wireformat/openwire/marshal/v6/**LastPartialCommandMarshaller.h**  
 4255  
 src/main/activemq/wireformat/openwire/marshal/v6/**LocalTransactionIdMarshaller.h**  
 4259  
 src/main/activemq/wireformat/openwire/marshal/v6/**MarshallerFactory.h** . . . 4263  
 src/main/activemq/wireformat/openwire/marshal/v6/**MessageAckMarshaller.h**  
 4267  
 src/main/activemq/wireformat/openwire/marshal/v6/**MessageDispatchMarshaller.h**  
 4271  
 src/main/activemq/wireformat/openwire/marshal/v6/**MessageDispatchNotificationMarshaller.h**  
 4276  
 src/main/activemq/wireformat/openwire/marshal/v6/**MessageIdMarshaller.h** . 4280  
 src/main/activemq/wireformat/openwire/marshal/v6/**MessageMarshaller.h** . . 4284  
 src/main/activemq/wireformat/openwire/marshal/v6/**MessagePullMarshaller.h**  
 4288  
 src/main/activemq/wireformat/openwire/marshal/v6/**NetworkBridgeFilterMarshaller.h**  
 4293  
 src/main/activemq/wireformat/openwire/marshal/v6/**PartialCommandMarshaller.h**  
 4297  
 src/main/activemq/wireformat/openwire/marshal/v6/**ProducerAckMarshaller.h**  
 4301  
 src/main/activemq/wireformat/openwire/marshal/v6/**ProducerIdMarshaller.h** . 4306  
 src/main/activemq/wireformat/openwire/marshal/v6/**ProducerInfoMarshaller.h**  
 4310  
 src/main/activemq/wireformat/openwire/marshal/v6/**RemoveInfoMarshaller.h**  
 4314  
 src/main/activemq/wireformat/openwire/marshal/v6/**RemoveSubscriptionInfoMarshaller.h**  
 4318  
 src/main/activemq/wireformat/openwire/marshal/v6/**ReplayCommandMarshaller.h**  
 4323  
 src/main/activemq/wireformat/openwire/marshal/v6/**ResponseMarshaller.h** . 4327  
 src/main/activemq/wireformat/openwire/marshal/v6/**SessionIdMarshaller.h** . 4331  
 src/main/activemq/wireformat/openwire/marshal/v6/**SessionInfoMarshaller.h**  
 4335  
 src/main/activemq/wireformat/openwire/marshal/v6/**ShutdownInfoMarshaller.h**  
 4339  
 src/main/activemq/wireformat/openwire/marshal/v6/**SubscriptionInfoMarshaller.h**  
 4344  
 src/main/activemq/wireformat/openwire/marshal/v6/**TransactionIdMarshaller.h**  
 4348  
 src/main/activemq/wireformat/openwire/marshal/v6/**TransactionInfoMarshaller.h**  
 4352  
 src/main/activemq/wireformat/openwire/marshal/v6/**WireFormatInfoMarshaller.h**  
 4357  
 src/main/activemq/wireformat/openwire/marshal/v6/**XATransactionIdMarshaller.h**  
 4361  
 src/main/activemq/wireformat/openwire/utils/**BooleanStream.h** . . . . . 4364  
 src/main/activemq/wireformat/openwire/utils/**HexTable.h** . . . . . 4365

src/main/activemq/wireformat/openwire/utls/ <b>MessagePropertyInterceptor.h</b>	4365
src/main/activemq/wireformat/stomp/ <b>StompCommandConstants.h</b>	4366
src/main/activemq/wireformat/stomp/ <b>StompFrame.h</b>	4366
src/main/activemq/wireformat/stomp/ <b>StompHelper.h</b>	4367
src/main/activemq/wireformat/stomp/ <b>StompWireFormat.h</b>	4368
src/main/activemq/wireformat/stomp/ <b>StompWireFormatFactory.h</b>	4368
src/main/cms/ <b>BytesMessage.h</b>	4371
src/main/cms/ <b>Closeable.h</b>	4372
src/main/cms/ <b>CMSException.h</b>	4373
src/main/cms/ <b>CMSProperties.h</b>	4373
src/main/cms/ <b>CMSSecurityException.h</b>	4374
src/main/cms/ <b>Config.h</b>	4087
src/main/cms/ <b>Connection.h</b>	4374
src/main/cms/ <b>ConnectionFactory.h</b>	4375
src/main/cms/ <b>ConnectionMetaData.h</b>	4375
src/main/cms/ <b>DeliveryMode.h</b>	4375
src/main/cms/ <b>Destination.h</b>	4376
src/main/cms/ <b>ExceptionListener.h</b>	4376
src/main/cms/ <b>IllegalStateException.h</b>	4377
src/main/cms/ <b>InvalidClientIdException.h</b>	4378
src/main/cms/ <b>InvalidDestinationException.h</b>	4378
src/main/cms/ <b>InvalidSelectorException.h</b>	4378
src/main/cms/ <b>MapMessage.h</b>	4379
src/main/cms/ <b>Message.h</b>	4027
src/main/cms/ <b>MessageConsumer.h</b>	4379
src/main/cms/ <b>MessageEnumeration.h</b>	4380
src/main/cms/ <b>MessageEOFException.h</b>	4380
src/main/cms/ <b>MessageFormatException.h</b>	4381
src/main/cms/ <b>MessageListener.h</b>	4381
src/main/cms/ <b>MessageNotReadableException.h</b>	4382
src/main/cms/ <b>MessageNotWritableException.h</b>	4382
src/main/cms/ <b>MessageProducer.h</b>	4382
src/main/cms/ <b>ObjectMessage.h</b>	4383
src/main/cms/ <b>Queue.h</b>	4383
src/main/cms/ <b>QueueBrowser.h</b>	4384
src/main/cms/ <b>Session.h</b>	4385
src/main/cms/ <b>Startable.h</b>	4386
src/main/cms/ <b>Stoppable.h</b>	4386
src/main/cms/ <b>StreamMessage.h</b>	4387
src/main/cms/ <b>TemporaryQueue.h</b>	4387
src/main/cms/ <b>TemporaryTopic.h</b>	4388
src/main/cms/ <b>TextMessage.h</b>	4388
src/main/cms/ <b>Topic.h</b>	4388
src/main/cms/ <b>UnsupportedOperationException.h</b>	4389
src/main/decaf/internal/ <b>AprPool.h</b>	4390
src/main/decaf/internal/ <b>DecafRuntime.h</b>	4390
src/main/decaf/internal/io/ <b>StandardErrorOutputStream.h</b>	4391
src/main/decaf/internal/io/ <b>StandardInputStream.h</b>	4391
src/main/decaf/internal/io/ <b>StandardOutputStream.h</b>	4392
src/main/decaf/internal/net/ <b>DefaultServerSocketFactory.h</b>	4392

src/main/decaf/internal/net/DefaultSocketFactory.h	4393
src/main/decaf/internal/net/Network.h	4393
src/main/decaf/internal/net/SocketFileDescriptor.h	4394
src/main/decaf/internal/net/URIEncoderDecoder.h	4402
src/main/decaf/internal/net/URIHelper.h	4403
src/main/decaf/internal/net/URIType.h	4403
src/main/decaf/internal/net/ssl/DefaultSSLContext.h	4394
src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h	4395
src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h	4395
src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h	4396
src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h	4396
src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h	4397
src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h	4397
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h	4398
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h	4398
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h	4399
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h	4399
src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h	4400
src/main/decaf/internal/net/tcp/TcpSocket.h	4400
src/main/decaf/internal/net/tcp/TcpSocketInputStream.h	4401
src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h	4402
src/main/decaf/internal/nio/BufferFactory.h	4404
src/main/decaf/internal/nio/ByteArrayBuffer.h	4404
src/main/decaf/internal/nio/CharArrayBuffer.h	4405
src/main/decaf/internal/nio/DoubleArrayBuffer.h	4406
src/main/decaf/internal/nio/FloatArrayBuffer.h	4406
src/main/decaf/internal/nio/IntArrayBuffer.h	4407
src/main/decaf/internal/nio/LongArrayBuffer.h	4408
src/main/decaf/internal/nio/ShortArrayBuffer.h	4408
src/main/decaf/internal/security/unix/SecureRandomImpl.h	4409
src/main/decaf/internal/security/windows/SecureRandomImpl.h	4409
src/main/decaf/internal/util/ByteArrayAdapter.h	4410
src/main/decaf/internal/util/GenericResource.h	4415
src/main/decaf/internal/util/HexStringParser.h	4416
src/main/decaf/internal/util/Resource.h	4416
src/main/decaf/internal/util/ResourceLifecycleManager.h	4000
src/main/decaf/internal/util/TimerTaskHeap.h	4417
src/main/decaf/internal/util/concurrent/ConditionImpl.h	4411
src/main/decaf/internal/util/concurrent/MutexImpl.h	4411
src/main/decaf/internal/util/concurrent/SynchronizableImpl.h	4412
src/main/decaf/internal/util/concurrent/Transferer.h	4412
src/main/decaf/internal/util/concurrent/TransferQueue.h	4413
src/main/decaf/internal/util/concurrent/TransferStack.h	4413
src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h	4414
src/main/decaf/internal/util/concurrent/unix/MutexHandle.h	4415
src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h	4414
src/main/decaf/internal/util/concurrent/windows/MutexHandle.h	4415
src/main/decaf/internal/util/zip/crc32.h	4417
src/main/decaf/internal/util/zip/deflate.h	4417
src/main/decaf/internal/util/zip/gzguts.h	4421

src/main/decaf/internal/util/zip/ <b>inffast.h</b>	4423
src/main/decaf/internal/util/zip/ <b>inffixed.h</b>	4423
src/main/decaf/internal/util/zip/ <b>inflate.h</b>	4423
src/main/decaf/internal/util/zip/ <b>inftrees.h</b>	4425
src/main/decaf/internal/util/zip/ <b>trees.h</b>	4426
src/main/decaf/internal/util/zip/ <b>zconf.h</b>	4428
src/main/decaf/internal/util/zip/ <b>zlib.h</b>	4430
src/main/decaf/internal/util/zip/ <b>zutil.h</b>	4437
src/main/decaf/io/ <b>BlockingByteArrayInputStream.h</b>	4440
src/main/decaf/io/ <b>BufferedInputStream.h</b>	4441
src/main/decaf/io/ <b>BufferedOutputStream.h</b>	4441
src/main/decaf/io/ <b>ByteArrayInputStream.h</b>	4442
src/main/decaf/io/ <b>ByteArrayOutputStream.h</b>	4442
src/main/decaf/io/ <b>Closeable.h</b>	4372
src/main/decaf/io/ <b>DataInput.h</b>	4443
src/main/decaf/io/ <b>DataInputStream.h</b>	4443
src/main/decaf/io/ <b>DataOutput.h</b>	4444
src/main/decaf/io/ <b>DataOutputStream.h</b>	4444
src/main/decaf/io/ <b>EOFException.h</b>	4445
src/main/decaf/io/ <b>FileDescriptor.h</b>	4445
src/main/decaf/io/ <b>FilterInputStream.h</b>	4446
src/main/decaf/io/ <b>FilterOutputStream.h</b>	4446
src/main/decaf/io/ <b>Flushable.h</b>	4447
src/main/decaf/io/ <b>InputStream.h</b>	4447
src/main/decaf/io/ <b>InputStreamReader.h</b>	4448
src/main/decaf/io/ <b>InterruptedIOException.h</b>	4448
src/main/decaf/io/ <b>IOException.h</b>	4449
src/main/decaf/io/ <b>OutputStream.h</b>	4449
src/main/decaf/io/ <b>OutputStreamWriter.h</b>	4450
src/main/decaf/io/ <b>PushbackInputStream.h</b>	4450
src/main/decaf/io/ <b>Reader.h</b>	4450
src/main/decaf/io/ <b>UnsupportedEncodingException.h</b>	4451
src/main/decaf/io/ <b>UTFDataFormatException.h</b>	4451
src/main/decaf/io/ <b>Writer.h</b>	4452
src/main/decaf/lang/ <b>Appendable.h</b>	4452
src/main/decaf/lang/ <b>ArrayPointer.h</b>	4453
src/main/decaf/lang/ <b>Boolean.h</b>	4454
src/main/decaf/lang/ <b>Byte.h</b>	4454
src/main/decaf/lang/ <b>Character.h</b>	4455
src/main/decaf/lang/ <b>CharSequence.h</b>	4455
src/main/decaf/lang/ <b>Comparable.h</b>	4456
src/main/decaf/lang/ <b>Double.h</b>	4456
src/main/decaf/lang/ <b>Exception.h</b>	4457
src/main/decaf/lang/ <b>Float.h</b>	4462
src/main/decaf/lang/ <b>Integer.h</b>	4462
src/main/decaf/lang/ <b>Iterable.h</b>	4463
src/main/decaf/lang/ <b>Long.h</b>	4463
src/main/decaf/lang/ <b>Math.h</b>	4464
src/main/decaf/lang/ <b>Number.h</b>	4464
src/main/decaf/lang/ <b>Pointer.h</b>	4465

src/main/decaf/lang/ <b>Readable.h</b>	4466
src/main/decaf/lang/ <b>Runnable.h</b>	4466
src/main/decaf/lang/ <b>Runtime.h</b>	4467
src/main/decaf/lang/ <b>Short.h</b>	4467
src/main/decaf/lang/ <b>String.h</b>	4468
src/main/decaf/lang/ <b>System.h</b>	4468
src/main/decaf/lang/ <b>Thread.h</b>	4469
src/main/decaf/lang/ <b>ThreadGroup.h</b>	4469
src/main/decaf/lang/ <b>Throwable.h</b>	4470
src/main/decaf/lang/exceptions/ <b>ClassCastException.h</b>	4457
src/main/decaf/lang/exceptions/ <b>ExceptionDefines.h</b>	4053
src/main/decaf/lang/exceptions/ <b>IllegalArgumentException.h</b>	4458
src/main/decaf/lang/exceptions/ <b>IllegalMonitorStateException.h</b>	4458
src/main/decaf/lang/exceptions/ <b>IllegalStateException.h</b>	4377
src/main/decaf/lang/exceptions/ <b>IllegalThreadStateException.h</b>	4459
src/main/decaf/lang/exceptions/ <b>IndexOutOfBoundsException.h</b>	4459
src/main/decaf/lang/exceptions/ <b>InterruptedException.h</b>	4459
src/main/decaf/lang/exceptions/ <b>InvalidStateException.h</b>	4460
src/main/decaf/lang/exceptions/ <b>NoSuchElementException.h</b>	4460
src/main/decaf/lang/exceptions/ <b>NullPointerException.h</b>	4461
src/main/decaf/lang/exceptions/ <b>NumberFormatException.h</b>	4461
src/main/decaf/lang/exceptions/ <b>RuntimeException.h</b>	4462
src/main/decaf/lang/exceptions/ <b>UnsupportedOperationException.h</b>	4389
src/main/decaf/net/ <b>BindException.h</b>	4470
src/main/decaf/net/ <b>ConnectException.h</b>	4471
src/main/decaf/net/ <b>HttpRetryException.h</b>	4471
src/main/decaf/net/ <b>Inet4Address.h</b>	4471
src/main/decaf/net/ <b>Inet6Address.h</b>	4472
src/main/decaf/net/ <b>InetAddress.h</b>	4472
src/main/decaf/net/ <b>InetSocketAddress.h</b>	4473
src/main/decaf/net/ <b>MalformedURLException.h</b>	4473
src/main/decaf/net/ <b>NoRouteToHostException.h</b>	4474
src/main/decaf/net/ <b>PortUnreachableException.h</b>	4474
src/main/decaf/net/ <b>ProtocolException.h</b>	4474
src/main/decaf/net/ <b>ServerSocket.h</b>	4475
src/main/decaf/net/ <b>ServerSocketFactory.h</b>	4475
src/main/decaf/net/ <b>Socket.h</b>	4476
src/main/decaf/net/ <b>SocketAddress.h</b>	4477
src/main/decaf/net/ <b>SocketError.h</b>	4477
src/main/decaf/net/ <b>SocketException.h</b>	4477
src/main/decaf/net/ <b>SocketFactory.h</b>	4478
src/main/decaf/net/ <b>SocketImpl.h</b>	4478
src/main/decaf/net/ <b>SocketImplFactory.h</b>	4479
src/main/decaf/net/ <b>SocketOptions.h</b>	4479
src/main/decaf/net/ <b>SocketTimeoutException.h</b>	4480
src/main/decaf/net/ <b>UnknownHostException.h</b>	4483
src/main/decaf/net/ <b>UnknownServiceException.h</b>	4484
src/main/decaf/net/ <b>URI.h</b>	4484
src/main/decaf/net/ <b>URISyntaxException.h</b>	4485
src/main/decaf/net/ <b>URL.h</b>	4485

src/main/decaf/net/ <b>URLDecoder.h</b> . . . . .	4486
src/main/decaf/net/ <b>URLEncoder.h</b> . . . . .	4486
src/main/decaf/net/ssl/ <b>SSLContext.h</b> . . . . .	4480
src/main/decaf/net/ssl/ <b>SSLContextSpi.h</b> . . . . .	4480
src/main/decaf/net/ssl/ <b>SSLParameters.h</b> . . . . .	4481
src/main/decaf/net/ssl/ <b>SSLServerSocket.h</b> . . . . .	4481
src/main/decaf/net/ssl/ <b>SSLServerSocketFactory.h</b> . . . . .	4482
src/main/decaf/net/ssl/ <b>SSLSocket.h</b> . . . . .	4482
src/main/decaf/net/ssl/ <b>SSLSocketFactory.h</b> . . . . .	4483
src/main/decaf/nio/ <b>Buffer.h</b> . . . . .	4486
src/main/decaf/nio/ <b>BufferOverflowException.h</b> . . . . .	4487
src/main/decaf/nio/ <b>BufferUnderflowException.h</b> . . . . .	4487
src/main/decaf/nio/ <b>ByteBuffer.h</b> . . . . .	4488
src/main/decaf/nio/ <b>CharBuffer.h</b> . . . . .	4488
src/main/decaf/nio/ <b>DoubleBuffer.h</b> . . . . .	4489
src/main/decaf/nio/ <b>FloatBuffer.h</b> . . . . .	4489
src/main/decaf/nio/ <b>IntBuffer.h</b> . . . . .	4490
src/main/decaf/nio/ <b>InvalidMarkException.h</b> . . . . .	4490
src/main/decaf/nio/ <b>LongBuffer.h</b> . . . . .	4491
src/main/decaf/nio/ <b>ReadOnlyBufferException.h</b> . . . . .	4491
src/main/decaf/nio/ <b>ShortBuffer.h</b> . . . . .	4492
src/main/decaf/security/ <b>GeneralSecurityException.h</b> . . . . .	4496
src/main/decaf/security/ <b>InvalidKeyException.h</b> . . . . .	4497
src/main/decaf/security/ <b>Key.h</b> . . . . .	4497
src/main/decaf/security/ <b>KeyException.h</b> . . . . .	4498
src/main/decaf/security/ <b>KeyManagementException.h</b> . . . . .	4498
src/main/decaf/security/ <b>NoSuchAlgorithmException.h</b> . . . . .	4498
src/main/decaf/security/ <b>NoSuchProviderException.h</b> . . . . .	4499
src/main/decaf/security/ <b>Principal.h</b> . . . . .	4499
src/main/decaf/security/ <b>PublicKey.h</b> . . . . .	4500
src/main/decaf/security/ <b>SecureRandom.h</b> . . . . .	4500
src/main/decaf/security/ <b>SecureRandomSpi.h</b> . . . . .	4501
src/main/decaf/security/ <b>SignatureException.h</b> . . . . .	4501
src/main/decaf/security/auth/x500/ <b>X500Principal.h</b> . . . . .	4492
src/main/decaf/security/cert/ <b>Certificate.h</b> . . . . .	4493
src/main/decaf/security/cert/ <b>CertificateEncodingException.h</b> . . . . .	4494
src/main/decaf/security/cert/ <b>CertificateException.h</b> . . . . .	4494
src/main/decaf/security/cert/ <b>CertificateExpiredException.h</b> . . . . .	4494
src/main/decaf/security/cert/ <b>CertificateNotYetValidException.h</b> . . . . .	4495
src/main/decaf/security/cert/ <b>CertificateParsingException.h</b> . . . . .	4495
src/main/decaf/security/cert/ <b>X509Certificate.h</b> . . . . .	4496
src/main/decaf/util/ <b>AbstractCollection.h</b> . . . . .	4501
src/main/decaf/util/ <b>AbstractList.h</b> . . . . .	4502
src/main/decaf/util/ <b>AbstractMap.h</b> . . . . .	4503
src/main/decaf/util/ <b>AbstractQueue.h</b> . . . . .	4503
src/main/decaf/util/ <b>AbstractSequentialList.h</b> . . . . .	4504
src/main/decaf/util/ <b>AbstractSet.h</b> . . . . .	4505
src/main/decaf/util/ <b>Collection.h</b> . . . . .	4505
src/main/decaf/util/ <b>Comparator.h</b> . . . . .	4506
src/main/decaf/util/ <b>Config.h</b> . . . . .	4087

src/main/decaf/util/ <b>Date.h</b>	4526
src/main/decaf/util/ <b>Iterator.h</b>	4526
src/main/decaf/util/ <b>List.h</b>	4526
src/main/decaf/util/ <b>ListIterator.h</b>	4527
src/main/decaf/util/ <b>Map.h</b>	4538
src/main/decaf/util/ <b>PriorityQueue.h</b>	4539
src/main/decaf/util/ <b>Properties.h</b>	4539
src/main/decaf/util/ <b>Queue.h</b>	4384
src/main/decaf/util/ <b>Random.h</b>	4540
src/main/decaf/util/ <b>Set.h</b>	4541
src/main/decaf/util/ <b>StlList.h</b>	4541
src/main/decaf/util/ <b>StlMap.h</b>	4542
src/main/decaf/util/ <b>StlQueue.h</b>	4542
src/main/decaf/util/ <b>StlSet.h</b>	4543
src/main/decaf/util/ <b>StringTokenizer.h</b>	4544
src/main/decaf/util/ <b>Timer.h</b>	4544
src/main/decaf/util/ <b>TimerTask.h</b>	4545
src/main/decaf/util/ <b>UUID.h</b>	4545
src/main/decaf/util/comparators/ <b>Less.h</b>	4506
src/main/decaf/util/concurrent/ <b>BlockingQueue.h</b>	4509
src/main/decaf/util/concurrent/ <b>BrokenBarrierException.h</b>	4509
src/main/decaf/util/concurrent/ <b>Callable.h</b>	4510
src/main/decaf/util/concurrent/ <b>CancellationException.h</b>	4510
src/main/decaf/util/concurrent/ <b>Concurrent.h</b>	4511
src/main/decaf/util/concurrent/ <b>ConcurrentMap.h</b>	4512
src/main/decaf/util/concurrent/ <b>ConcurrentStlMap.h</b>	4512
src/main/decaf/util/concurrent/ <b>CountDownLatch.h</b>	4513
src/main/decaf/util/concurrent/ <b>Delayed.h</b>	4513
src/main/decaf/util/concurrent/ <b>ExecutionException.h</b>	4514
src/main/decaf/util/concurrent/ <b>Executor.h</b>	4514
src/main/decaf/util/concurrent/ <b>ExecutorService.h</b>	4515
src/main/decaf/util/concurrent/ <b>Future.h</b>	4515
src/main/decaf/util/concurrent/ <b>Lock.h</b>	4516
src/main/decaf/util/concurrent/ <b>Mutex.h</b>	4519
src/main/decaf/util/concurrent/ <b>PooledThread.h</b>	4519
src/main/decaf/util/concurrent/ <b>PooledThreadListener.h</b>	4520
src/main/decaf/util/concurrent/ <b>RejectedExecutionException.h</b>	4520
src/main/decaf/util/concurrent/ <b>RejectedExecutionHandler.h</b>	4521
src/main/decaf/util/concurrent/ <b>Semaphore.h</b>	4521
src/main/decaf/util/concurrent/ <b>Synchronizable.h</b>	4522
src/main/decaf/util/concurrent/ <b>SynchronousQueue.h</b>	4522
src/main/decaf/util/concurrent/ <b>TaskListener.h</b>	4523
src/main/decaf/util/concurrent/ <b>ThreadFactory.h</b>	4523
src/main/decaf/util/concurrent/ <b>ThreadPool.h</b>	4524
src/main/decaf/util/concurrent/ <b>TimeoutException.h</b>	4524
src/main/decaf/util/concurrent/ <b>TimeUnit.h</b>	4525
src/main/decaf/util/concurrent/atomic/ <b>AtomicBoolean.h</b>	4507
src/main/decaf/util/concurrent/atomic/ <b>AtomicInteger.h</b>	4507
src/main/decaf/util/concurrent/atomic/ <b>AtomicRefCounter.h</b>	4508
src/main/decaf/util/concurrent/atomic/ <b>AtomicReference.h</b>	4508



src/main/decaf/util/concurrent/locks/ <b>Condition.h</b>	4517
src/main/decaf/util/concurrent/locks/ <b>Lock.h</b>	4516
src/main/decaf/util/concurrent/locks/ <b>LockSupport.h</b>	4517
src/main/decaf/util/concurrent/locks/ <b>ReadWriteLock.h</b>	4518
src/main/decaf/util/concurrent/locks/ <b>ReentrantLock.h</b>	4518
src/main/decaf/util/logging/ <b>ConsoleHandler.h</b>	4528
src/main/decaf/util/logging/ <b>ErrorManager.h</b>	4528
src/main/decaf/util/logging/ <b>Filter.h</b>	4529
src/main/decaf/util/logging/ <b>Formatter.h</b>	4529
src/main/decaf/util/logging/ <b>Handler.h</b>	4529
src/main/decaf/util/logging/ <b>Level.h</b>	4530
src/main/decaf/util/logging/ <b>Logger.h</b>	4531
src/main/decaf/util/logging/ <b>LoggerCommon.h</b>	4531
src/main/decaf/util/logging/ <b>LoggerDefines.h</b>	4532
src/main/decaf/util/logging/ <b>LoggerHierarchy.h</b>	4533
src/main/decaf/util/logging/ <b>LogManager.h</b>	4533
src/main/decaf/util/logging/ <b>LogRecord.h</b>	4534
src/main/decaf/util/logging/ <b>LogWriter.h</b>	4535
src/main/decaf/util/logging/ <b>MarkBlockLogger.h</b>	4535
src/main/decaf/util/logging/ <b>PropertiesChangeListener.h</b>	4536
src/main/decaf/util/logging/ <b>SimpleFormatter.h</b>	4536
src/main/decaf/util/logging/ <b>SimpleLogger.h</b>	4537
src/main/decaf/util/logging/ <b>StreamHandler.h</b>	4537
src/main/decaf/util/logging/ <b>XMLFormatter.h</b>	4538
src/main/decaf/util/zip/ <b>Adler32.h</b>	4546
src/main/decaf/util/zip/ <b>CheckedInputStream.h</b>	4546
src/main/decaf/util/zip/ <b>CheckedOutputStream.h</b>	4547
src/main/decaf/util/zip/ <b>Checksum.h</b>	4547
src/main/decaf/util/zip/ <b>CRC32.h</b>	4548
src/main/decaf/util/zip/ <b>DataFormatException.h</b>	4548
src/main/decaf/util/zip/ <b>Deflater.h</b>	4549
src/main/decaf/util/zip/ <b>DeflaterOutputStream.h</b>	4549
src/main/decaf/util/zip/ <b>Inflater.h</b>	4550
src/main/decaf/util/zip/ <b>InflaterInputStream.h</b>	4550
src/main/decaf/util/zip/ <b>ZipException.h</b>	4551



## Chapter 5

# Namespace Documentation

### 5.1 activemq Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

#### Namespaces

- namespace **cmsutil**
- namespace **commands**
- namespace **core**
- namespace **exceptions**
- namespace **io**
- namespace **library**
- namespace **state**
- namespace **threads**
- namespace **transport**
- namespace **util**
- namespace **wireformat**

#### 5.1.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.2 activemq::cmsutil Namespace Reference

### Data Structures

- class **CachedConsumer**  
*A cached message consumer contained within a pooled session.*
- class **CachedProducer**  
*A cached message producer contained within a pooled session.*
- class **CmsAccessor**  
*Base class for **activemq.cmsutil.CmsTemplate** (p. 1140) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1294) to operate on.*
- class **CmsDestinationAccessor**  
*Extends the **CmsAccessor** (p. 1123) to add support for resolving destination names.*
- class **CmsTemplate**  
***CmsTemplate** (p. 1140) simplifies performing synchronous CMS operations.*
- class **DestinationResolver**  
*Resolves a CMS destination name to a *Destination*.*
- class **DynamicDestinationResolver**  
*Resolves a CMS destination name to a *Destination*.*
- class **MessageCreator**  
*Creates the user-defined message to be sent by the **CmsTemplate** (p. 1140).*
- class **PooledSession**  
*A pooled session object that wraps around a delegate session.*
- class **ProducerCallback**  
*Callback for sending a message to a CMS destination.*
- class **ResourceLifecycleManager**  
*Manages the lifecycle of a set of CMS resources.*
- class **SessionCallback**  
*Callback for executing any number of operations on a provided CMS Session.*
- class **SessionPool**  
*A pool of CMS sessions from the same connection and with the same acknowledge mode.*

## 5.3 activemq::commands Namespace Reference

### Data Structures

- class **ActiveMQBlobMessage**
- class **ActiveMQBytesMessage**
- class **ActiveMQDestination**
- class **ActiveMQMapMessage**
- class **ActiveMQMessage**
- class **ActiveMQMessageTemplate**
- class **ActiveMQObjectMessage**
- class **ActiveMQQueue**
- class **ActiveMQStreamMessage**
- class **ActiveMQTempDestination**
- class **ActiveMQTempQueue**
- class **ActiveMQTempTopic**
- class **ActiveMQTextMessage**
- class **ActiveMQTopic**
- class **BaseCommand**
- class **BaseDataStructure**
- class **BooleanExpression**
- class **BrokerError**

*This class represents an Exception sent from the Broker.*

- class **BrokerId**
- class **BrokerInfo**
- class **Command**
- class **ConnectionControl**
- class **ConnectionError**
- class **ConnectionId**
- class **ConnectionInfo**
- class **ConsumerControl**
- class **ConsumerId**
- class **ConsumerInfo**
- class **ControlCommand**
- class **DataArrayResponse**
- class **DataResponse**
- class **DataStructure**
- class **DestinationInfo**
- class **DiscoveryEvent**
- class **ExceptionResponse**
- class **FlushCommand**
- class **IntegerResponse**
- class **JournalQueueAck**
- class **JournalTopicAck**
- class **JournalTrace**
- class **JournalTransaction**

- class **KeepAliveInfo**
- class **LastPartialCommand**
- class **LocalTransactionId**
- class **Message**
- class **MessageAck**
- class **MessageDispatch**
- class **MessageDispatchNotification**
- class **MessageId**
- class **MessagePull**
- class **NetworkBridgeFilter**
- class **PartialCommand**
- class **ProducerAck**
- class **ProducerId**
- class **ProducerInfo**
- class **RemoveInfo**
- class **RemoveSubscriptionInfo**
- class **ReplayCommand**
- class **Response**
- class **SessionId**
- class **SessionInfo**
- class **ShutdownInfo**
- class **SubscriptionInfo**
- class **TransactionId**
- class **TransactionInfo**
- class **WireFormatInfo**
- class **XATransactionId**

## 5.4 activemq::core Namespace Reference

### Namespaces

- namespace **policies**

### Data Structures

- class **ActiveMQAckHandler**  
*Interface class that is used to give CMS Messages an interface to Ack themselves with.*
- class **ActiveMQConnection**  
*Concrete connection used for all connectors to the ActiveMQ broker.*
- class **ActiveMQConnectionFactory**
- class **ActiveMQConnectionMetaData**  
*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 244) class.*

- class **ActiveMQConstants**  
*Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.*
- class **ActiveMQConsumer**
- class **ActiveMQProducer**
- class **ActiveMQQueueBrowser**
- class **ActiveMQSession**
- class **ActiveMQSessionExecutor**  
*Delegate dispatcher for a single session.*
- class **ActiveMQTransactionContext**  
*Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.*
- class **DispatchData**  
*Simple POCO that contains the information necessary to route a message to a specified consumer.*
- class **Dispatcher**  
*Interface for an object responsible for dispatching messages to consumers.*
- class **MessageDispatchChannel**
- class **PrefetchPolicy**  
*Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.*
- class **RedeliveryPolicy**  
*Interface for a **RedeliveryPolicy** (p. 3121) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*
- class **Synchronization**  
*Transacted Object **Synchronization** (p. 3659), used to sync the events of a Transaction with the items in the Transaction.*

## 5.5 activemq::core::policies Namespace Reference

### Data Structures

- class **DefaultPrefetchPolicy**
- class **DefaultRedeliveryPolicy**

## 5.6 activemq::exceptions Namespace Reference

### Data Structures

- class **ActiveMQException**
- class **BrokerException**

## 5.7 activemq::io Namespace Reference

### Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**  
*OutputStream filter that just logs the data being written.*

## 5.8 activemq::library Namespace Reference

### Data Structures

- class **ActiveMQCPP**

## 5.9 activemq::state Namespace Reference

### Data Structures

- class **CommandVisitor**  
*Interface for an Object that can visit the various Command Objects that are sent from and to this client.*
- class **CommandVisitorAdapter**  
*Default Implementation of a **CommandVisitor** (p. 1171) that returns NULL for all calls.*
- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

## 5.10 activemq::threads Namespace Reference

### Data Structures

- class **CompositeTask**  
*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 1194).*
- class **CompositeTaskRunner**  
*A **Task** (p. 3678) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*



- class **DedicatedTaskRunner**
- class **Task**  
*Represents a unit of work that requires one or more iterations to complete.*
- class **TaskRunner**

## 5.11 activemq::transport Namespace Reference

### Namespaces

- namespace **correlator**
- namespace **failover**
- namespace **inactivity**
- namespace **logging**
- namespace **mock**
- namespace **tcp**

### Data Structures

- class **AbstractTransportFactory**  
*Abstract implementation of the **TransportFactory** (p. 3825) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3825) instances.*
- class **CompositeTransport**  
*A Composite **Transport** (p. 3819) is a **Transport** (p. 3819) implementation that is composed of several Transports.*
- class **DefaultTransportListener**
- class **IOTransport**  
*Implementation of the **Transport** (p. 3819) interface that performs marshaling of commands to IO streams.*
- class **Transport**  
*Interface for a transport layer for command objects.*
- class **TransportFactory**  
*Defines the interface for Factories that create Transports or TransportFilters.*
- class **TransportFilter**  
*A filter on the transport layer.*
- class **TransportListener**  
*A listener of asynchronous exceptions from a command transport object.*
- class **TransportRegistry**  
*Registry of all **Transport** (p. 3819) Factories that are available to the client at runtime.*

## 5.12 activemq::transport::correlator Namespace Reference

### Data Structures

- class **FutureResponse**  
*A container that holds a response object.*
- class **ResponseCorrelator**  
*This type of transport filter is responsible for correlating asynchronous responses with requests.*

## 5.13 activemq::transport::failover Namespace Reference

### Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**
- class **FailoverTransportFactory**  
*Creates an instance of a **FailoverTransport** (p. 1835).*
- class **FailoverTransportListener**  
*Utility class used by the **Transport** (p. 3819) to perform the work of responding to events from the active **Transport** (p. 3819).*
- class **URIPool**

## 5.14 activemq::transport::inactivity Namespace Reference

### Data Structures

- class **InactivityMonitor**
- class **ReadChecker**  
*Runnable class that is used by the {.*
- class **WriteChecker**  
*Runnable class used by the {.*

## 5.15 activemq::transport::logging Namespace Reference

### Data Structures

- class **LoggingTransport**  
*A transport filter that logs commands as they are sent/received.*

## 5.16 activemq::transport::mock Namespace Reference

### Data Structures

- class **InternalCommandListener**  
*Listens for Commands sent from the **MockTransport** (p. 2724).*
- class **MockTransport**  
*The **MockTransport** (p. 2724) defines a base level **Transport** (p. 3819) class that is intended to be used in place of an a regular protocol **Transport** (p. 3819) such as TCP.*
- class **MockTransportFactory**  
*Manufactures MockTransports, which are objects that read from input streams and write to output streams.*
- class **ResponseBuilder**  
*Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.*

## 5.17 activemq::transport::tcp Namespace Reference

### Data Structures

- class **SslTransport**  
***Transport** (p. 3819) for connecting to a Broker using an SSL Socket.*
- class **SslTransportFactory**
- class **TcpTransport**  
*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2105).*
- class **TcpTransportFactory**  
*Factory Responsible for creating the **TcpTransport** (p. 3696).*

## 5.18 activemq::util Namespace Reference

### Data Structures

- class **ActiveMQProperties**  
*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3072) object.*
- class **CMSExceptionSupport**
- class **CompositeData**  
*Represents a Composite URI.*
- class **IdGenerator**
- class **LongSequenceGenerator**  
*This class is used to generate a sequence of long long values that are incremented each time a new value is requested.*

- class **MarshallingSupport**
- class **MemoryUsage**
- class **PrimitiveList**  
*List of primitives.*
- class **PrimitiveMap**  
*Map of named primitives.*
- class **PrimitiveValueConverter**  
*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2960) from one type to another.*
- class **PrimitiveValueNode**  
*Class that wraps around a single value of one of the many types.*
- class **URISupport**
- class **Usage**

## 5.19 activemq::wireformat Namespace Reference

### Namespaces

- namespace **openwire**
- namespace **stomp**

### Data Structures

- class **MarshalAware**
- class **WireFormat**  
*Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.*
- class **WireFormatFactory**  
*The **WireFormatFactory** (p. 3911) is the interface that all **WireFormatFactory** (p. 3911) classes must extend.*
- class **WireFormatNegotiator**  
*Defines a **WireFormatNegotiator** (p. 3946) which allows a **WireFormat** (p. 3907) to.*
- class **WireFormatRegistry**  
*Registry of all **WireFormat** (p. 3907) Factories that are available to the client at run-time.*

## 5.20 activemq::wireformat::openwire Namespace Reference

### Namespaces

- namespace **marshal**
- namespace **utils**

## Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**

## 5.21 activemq::wireformat::openwire::marshal Namespace Reference

### Namespaces

- namespace **v1**
- namespace **v2**
- namespace **v3**
- namespace **v4**
- namespace **v5**
- namespace **v6**

## Data Structures

- class **BaseDataStreamMarshaller**  
*Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.*
- class **DataStreamMarshaller**  
*Base class for all classes that marshal commands for Openwire.*
- class **PrimitiveTypesMarshaller**  
*This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

## 5.22 activemq::wireformat::openwire::marshal::v1 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 182).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 224).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 308).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 348).*

- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 375).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 421).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 464).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 527).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 555).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 582).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 615).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 644).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 672).*
- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 743).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 840).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 871).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1250).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1282).*
- class **ConnectionIdMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1313).*
- class **ConnectionInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1343).*
- class **ConsumerControlMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1386).*
- class **ConsumerIdMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1414).*
- class **ConsumerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1447).*
- class **ControlCommandMarshaller**  
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1475).*
- class **DataArrayResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1508).*

- class **DataResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1573).*
- class **DestinationInfoMarshaller**  
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1708).*
- class **DiscoveryEventMarshaller**  
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1741).*
- class **ExceptionResponseMarshaller**  
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1825).*
- class **FlushCommandMarshaller**  
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1919).*
- class **IntegerResponseMarshaller**  
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2073).*
- class **JournalQueueAckMarshaller**  
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2139).*
- class **JournalTopicAckMarshaller**  
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2168).*
- class **JournalTraceMarshaller**  
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2190).*
- class **JournalTransactionMarshaller**  
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2221).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2249).*
- class **LastPartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2283).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2330).*
- class **MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2542).*
- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2582).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2611).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2648).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2670).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2716).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2769).*

- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2891).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3008).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3039).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3056).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3153).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3169).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3201).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3255).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3344).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3360).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3424).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3624).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3766).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3793).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3939).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3976).*

## 5.23 activemq::wireformat::openwire::marshal::v2 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 190).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 240).*



- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 320).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 360).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 387).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 433).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 476).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 539).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 566).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 594).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 623).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 656).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 684).*
- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 764).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 852).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 883).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1262).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1270).*
- class **ConnectionIdMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1301).*
- class **ConnectionInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1330).*
- class **ConsumerControlMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1373).*
- class **ConsumerIdMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1402).*
- class **ConsumerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1434).*

- class **ControlCommandMarshaller**  
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1462).*
- class **DataArrayResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1496).*
- class **DataResponseMarshaller**  
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1561).*
- class **DestinationInfoMarshaller**  
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1696).*
- class **DiscoveryEventMarshaller**  
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1729).*
- class **ExceptionResponseMarshaller**  
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1809).*
- class **FlushCommandMarshaller**  
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1907).*
- class **IntegerResponseMarshaller**  
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2061).*
- class **JournalQueueAckMarshaller**  
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2123).*
- class **JournalTopicAckMarshaller**  
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2152).*
- class **JournalTraceMarshaller**  
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2174).*
- class **JournalTransactionMarshaller**  
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2205).*
- class **KeepAliveInfoMarshaller**  
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2233).*
- class **LastPartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2271).*
- class **LocalTransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2314).*
- class **MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**  
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2530).*
- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2566).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2599).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2628).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2661).*

- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2700).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2749).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2874).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2988).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3019).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3052).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3141).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3178).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3205).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3241).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3324).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3368).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3420).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3640).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3770).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3809).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3931).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3968).*

## 5.24 activemq::wireformat::openwire::marshal::v3 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 177).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 220).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 304).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 344).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 371).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 416).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 460).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 523).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 551).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 578).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 607).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 635).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 664).*
- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 730).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 832).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 862).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1242).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1274).*
- class **ConnectionIdMarshaller**

- Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1305).*
- class **ConnectionInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1335).*
- class **ConsumerControlMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1378).*
- class **ConsumerIdMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1406).*
- class **ConsumerInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1439).*
- class **ControlCommandMarshaller**
  - Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1467).*
- class **DataArrayResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1500).*
- class **DataResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1565).*
- class **DestinationInfoMarshaller**
  - Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1700).*
- class **DiscoveryEventMarshaller**
  - Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1733).*
- class **ExceptionResponseMarshaller**
  - Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1813).*
- class **FlushCommandMarshaller**
  - Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1911).*
- class **IntegerResponseMarshaller**
  - Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2065).*
- class **JournalQueueAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2131).*
- class **JournalTopicAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2156).*
- class **JournalTraceMarshaller**
  - Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2178).*
- class **JournalTransactionMarshaller**
  - Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2209).*
- class **KeepAliveInfoMarshaller**
  - Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2237).*
- class **LastPartialCommandMarshaller**
  - Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2267).*
- class **LocalTransactionIdMarshaller**
  - Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2318).*
- class **MarshallerFactory**
  - Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**
  - Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2534).*

- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2570).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2603).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2640).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2657).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2708).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2761).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2883).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2996).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3027).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3064).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3149).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3174).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3209).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3250).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3340).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3364).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3432).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3620).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3774).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3797).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3943).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3980).*

## 5.25 activemq::wireformat::openwire::marshal::v4 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 186).
- class **ActiveMQBytesMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 228).
- class **ActiveMQDestinationMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 312).
- class **ActiveMQMapMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 352).
- class **ActiveMQMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 379).
- class **ActiveMQObjectMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 425).
- class **ActiveMQQueueMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 468).
- class **ActiveMQStreamMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 531).
- class **ActiveMQTempDestinationMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 558).
- class **ActiveMQTempQueueMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 586).
- class **ActiveMQTempTopicMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 611).
- class **ActiveMQTextMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 640).
- class **ActiveMQTopicMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 668).
- class **BaseCommandMarshaller**  
Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 737).
- class **BrokerIdMarshaller**  
Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 836).
- class **BrokerInfoMarshaller**  
Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 867).
- class **ConnectionControlMarshaller**  
Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1246).
- class **ConnectionErrorMarshaller**  
Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1278).
- class **ConnectionIdMarshaller**

- Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1309).*
- class **ConnectionInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1339).*
- class **ConsumerControlMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1382).*
- class **ConsumerIdMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1410).*
- class **ConsumerInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1443).*
- class **ControlCommandMarshaller**
  - Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1471).*
- class **DataArrayResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1504).*
- class **DataResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1569).*
- class **DestinationInfoMarshaller**
  - Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1704).*
- class **DiscoveryEventMarshaller**
  - Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1737).*
- class **ExceptionResponseMarshaller**
  - Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1821).*
- class **FlushCommandMarshaller**
  - Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1915).*
- class **IntegerResponseMarshaller**
  - Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2069).*
- class **JournalQueueAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2135).*
- class **JournalTopicAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2164).*
- class **JournalTraceMarshaller**
  - Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2186).*
- class **JournalTransactionMarshaller**
  - Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2217).*
- class **KeepAliveInfoMarshaller**
  - Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2241).*
- class **LastPartialCommandMarshaller**
  - Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2279).*
- class **LocalTransactionIdMarshaller**
  - Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2326).*
- class **MarshallerFactory**
  - Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**
  - Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2538).*



- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2578).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2607).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2632).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2666).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2712).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2765).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2887).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2992).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3023).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3047).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3161).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3190).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3197).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3236).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3328).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3372).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3436).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3632).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3778).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3805).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3935).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3972).*

## 5.26 activemq::wireformat::openwire::marshal::v5 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 194).*
- class **ActiveMQBytesMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 232).*
- class **ActiveMQDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 316).*
- class **ActiveMQMapMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 356).*
- class **ActiveMQMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 383).*
- class **ActiveMQObjectMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 429).*
- class **ActiveMQQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 472).*
- class **ActiveMQStreamMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 535).*
- class **ActiveMQTempDestinationMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 562).*
- class **ActiveMQTempQueueMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 590).*
- class **ActiveMQTempTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 619).*
- class **ActiveMQTextMessageMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 648).*
- class **ActiveMQTopicMarshaller**  
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 676).*
- class **BaseCommandMarshaller**  
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 750).*
- class **BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 844).*
- class **BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 875).*
- class **ConnectionControlMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1254).*
- class **ConnectionErrorMarshaller**  
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1286).*
- class **ConnectionIdMarshaller**

- Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1317).*
- class **ConnectionInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1347).*
- class **ConsumerControlMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1390).*
- class **ConsumerIdMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1418).*
- class **ConsumerInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1451).*
- class **ControlCommandMarshaller**
  - Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1479).*
- class **DataArrayResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1512).*
- class **DataResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1553).*
- class **DestinationInfoMarshaller**
  - Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1716).*
- class **DiscoveryEventMarshaller**
  - Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1745).*
- class **ExceptionResponseMarshaller**
  - Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1817).*
- class **FlushCommandMarshaller**
  - Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1923).*
- class **IntegerResponseMarshaller**
  - Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2077).*
- class **JournalQueueAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2127).*
- class **JournalTopicAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2148).*
- class **JournalTraceMarshaller**
  - Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2194).*
- class **JournalTransactionMarshaller**
  - Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2213).*
- class **KeepAliveInfoMarshaller**
  - Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2245).*
- class **LastPartialCommandMarshaller**
  - Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2275).*
- class **LocalTransactionIdMarshaller**
  - Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2322).*
- class **MarshallerFactory**
  - Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**
  - Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2546).*

- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2574).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2616).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2636).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2653).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2704).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2757).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2878).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3000).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3031).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3060).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3157).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3186).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3217).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3246).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3336).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3356).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3428).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3628).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3763).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3789).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3923).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3984).*

## 5.27 activemq::wireformat::openwire::marshal::v6 Namespace Reference

### Data Structures

- class **ActiveMQBlobMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 198).
- class **ActiveMQBytesMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 236).
- class **ActiveMQDestinationMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 324).
- class **ActiveMQMapMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 364).
- class **ActiveMQMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 391).
- class **ActiveMQObjectMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 437).
- class **ActiveMQQueueMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 480).
- class **ActiveMQStreamMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 543).
- class **ActiveMQTempDestinationMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 570).
- class **ActiveMQTempQueueMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 598).
- class **ActiveMQTempTopicMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 627).
- class **ActiveMQTextMessageMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 652).
- class **ActiveMQTopicMarshaller**  
Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 680).
- class **BaseCommandMarshaller**  
Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 757).
- class **BrokerIdMarshaller**  
Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 848).
- class **BrokerInfoMarshaller**  
Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 879).
- class **ConnectionControlMarshaller**  
Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1258).
- class **ConnectionErrorMarshaller**  
Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1290).
- class **ConnectionIdMarshaller**

- Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1321).*
- class **ConnectionInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1351).*
- class **ConsumerControlMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1394).*
- class **ConsumerIdMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1422).*
- class **ConsumerInfoMarshaller**
  - Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1455).*
- class **ControlCommandMarshaller**
  - Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1483).*
- class **DataArrayResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1516).*
- class **DataResponseMarshaller**
  - Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1557).*
- class **DestinationInfoMarshaller**
  - Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1712).*
- class **DiscoveryEventMarshaller**
  - Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1725).*
- class **ExceptionResponseMarshaller**
  - Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1804).*
- class **FlushCommandMarshaller**
  - Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1903).*
- class **IntegerResponseMarshaller**
  - Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2057).*
- class **JournalQueueAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2119).*
- class **JournalTopicAckMarshaller**
  - Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2160).*
- class **JournalTraceMarshaller**
  - Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2182).*
- class **JournalTransactionMarshaller**
  - Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2201).*
- class **KeepAliveInfoMarshaller**
  - Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2228).*
- class **LastPartialCommandMarshaller**
  - Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2262).*
- class **LocalTransactionIdMarshaller**
  - Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2310).*
- class **MarshallerFactory**
  - Used to create marshallers for a specific version of the wire protocol.*
- class **MessageAckMarshaller**
  - Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2526).*

- class **MessageDispatchMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2586).*
- class **MessageDispatchNotificationMarshaller**  
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2595).*
- class **MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2644).*
- class **MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2674).*
- class **MessagePullMarshaller**  
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2720).*
- class **NetworkBridgeFilterMarshaller**  
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2753).*
- class **PartialCommandMarshaller**  
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2870).*
- class **ProducerAckMarshaller**  
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3004).*
- class **ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3035).*
- class **ProducerInfoMarshaller**  
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3068).*
- class **RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3145).*
- class **RemoveSubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3182).*
- class **ReplayCommandMarshaller**  
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3213).*
- class **ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3260).*
- class **SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3332).*
- class **SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3352).*
- class **ShutdownInfoMarshaller**  
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3416).*
- class **SubscriptionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3636).*
- class **TransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3781).*
- class **TransactionInfoMarshaller**  
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3801).*
- class **WireFormatInfoMarshaller**  
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3927).*
- class **XATransactionIdMarshaller**  
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3964).*

## 5.28 activemq::wireformat::openwire::utils Namespace Reference

### Data Structures

- class **BooleanStream**  
*Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.*
- class **HexTable**  
*The **HexTable** (p. 1947) class maps hexadecimal strings to the value of an index into the table, i.e.*
- class **MessagePropertyInterceptor**  
*Used the base `ActiveMQMessage` class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the `OpenWire` Message properties.*

## 5.29 activemq::wireformat::stomp Namespace Reference

### Data Structures

- class **StompCommandConstants**
- class **StompFrame**  
*A Stomp-level message frame that encloses all messages to and from the broker.*
- class **StompHelper**  
*Utility Methods used when marshaling to and from `StompFrame`'s.*
- class **StompWireFormat**
- class **StompWireFormatFactory**  
*Factory used to create the `Stomp Wire Format` instance.*

## 5.30 cms Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

### Data Structures

- class **BytesMessage**  
*A **BytesMessage** (p. 1023) object is used to send a message containing a stream of unsigned bytes.*
- class **Closeable**  
*Interface for a class that implements the close method.*
- class **CMSException**  
*CMS API Exception that is the base for all exceptions thrown from CMS classes.*



- class **CMSProperties**  
*Interface for a Java-like properties object.*
- class **CMSSecurityException**  
*This exception must be thrown when a provider rejects a user name/password submitted by a client.*
- class **Connection**  
*The client's connection to its provider.*
- class **ConnectionFactory**  
*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1232) objects returned implement the CMS **Connection** (p. 1232) interface and hide the CMS Provider specific implementation details behind that interface.*
- class **ConnectionMetaData**  
*A **ConnectionMetaData** (p. 1355) object provides information describing the **Connection** (p. 1232) object.*
- class **DeliveryMode**  
*This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.*
- class **Destination**  
*A **Destination** (p. 1688) object encapsulates a provider-specific address.*
- class **ExceptionListener**  
*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1801) that is registered with the **Connection** (p. 1232).*
- class **IllegalStateException**  
*This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.*
- class **InvalidClientIdException**  
*This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.*
- class **InvalidDestinationException**  
*This exception must be thrown when a destination either is not understood by a provider or is no longer valid.*
- class **InvalidSelectorException**  
*This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.*
- class **MapMessage**  
*A **MapMessage** (p. 2431) object is used to send a set of name-value pairs.*
- class **Message**  
*Root of all messages.*
- class **MessageConsumer**  
*A client uses a **MessageConsumer** (p. 2550) to received messages from a destination.*
- class **MessageEnumeration**  
*Defines an object that enumerates a collection of Messages.*
- class **MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3595) or **BytesMessage** (p. 1023) is being read.*

- class **MessageFormatException**

*This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.*

- class **MessageListener**

*A **MessageListener** (p. 2652) object is used to receive asynchronously delivered messages.*

- class **MessageNotReadableException**

*This exception must be thrown when a CMS client attempts to read a write-only message.*

- class **MessageNotWriteableException**

*This exception must be thrown when a CMS client attempts to write to a read-only message.*

- class **MessageProducer**

*A client uses a **MessageProducer** (p. 2681) object to send messages to a **Destination** (p. 1688).*

- class **ObjectMessage**

*Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.*

- class **Queue**

*An interface encapsulating a provider-specific queue name.*

- class **QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p. 3093) without removing them.*

- class **Session**

*A **Session** (p. 3305) object is a single-threaded context for producing and consuming messages.*

- class **Startable**

*Interface for a class that implements the start method.*

- class **Stoppable**

*Interface for a class that implements the stop method.*

- class **StreamMessage**

*Interface for a **StreamMessage** (p. 3595).*

- class **TemporaryQueue**

*Defines a Temporary **Queue** (p. 3093) based **Destination** (p. 1688).*

- class **TemporaryTopic**

*Defines a Temporary **Topic** (p. 3757) based **Destination** (p. 1688).*

- class **TextMessage**

*Interface for a text message.*

- class **Topic**

*An interface encapsulating a provider-specific topic name.*

- class **UnsupportedOperationException**

*This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.*

### 5.30.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.31 decaf Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

### Namespaces

- namespace **internal**
- namespace **io**
- namespace **lang**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

### 5.31.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## 5.32 decaf::internal Namespace Reference

## Namespaces

- namespace **io**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **util**

## Data Structures

- class **AprPool**  
*Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.*
- class **DecafRuntime**  
*Handles APR initialization and termination.*

## 5.33 decaf::internal::io Namespace Reference

### Data Structures

- class **StandardErrorOutputStream**  
*Wrapper Around the Standard error Output facility on the current platform.*
- class **StandardInputStream**
- class **StandardOutputStream**

## 5.34 decaf::internal::net Namespace Reference

## Namespaces

- namespace **ssl**
- namespace **tcp**

## Data Structures

- class **DefaultServerSocketFactory**  
*Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.*
- class **DefaultSocketFactory**  
*SocketFactory implementation that is used to create Sockets.*
- class **Network**  
*Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.*

- class **SocketFileDescriptor**  
*File Descriptor type used internally by Decaf Socket objects.*
- class **URIEncoderDecoder**
- class **URIHelper**  
*Helper class used by the URI classes in encoding and decoding of URI's.*
- class **URIType**  
*Basic type object that holds data that composes a given URI.*

## 5.35 decaf::internal::net::ssl Namespace Reference

### Namespaces

- namespace **openssl**

### Data Structures

- class **DefaultSSLContext**  
*Default SSLContext manager for the Decaf library.*
- class **DefaultSSLServerSocketFactory**  
*Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*
- class **DefaultSSLSocketFactory**  
*Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

## 5.36 decaf::internal::net::ssl::openssl Namespace Reference

### Data Structures

- class **OpenSSLContextSpi**  
*Provides an SSLContext that wraps the OpenSSL API.*
- class **OpenSSLParameters**  
*Container class for parameters that are Common to OpenSSL socket classes.*
- class **OpenSSLServerSocket**  
*SSLServerSocket based on OpenSSL library code.*
- class **OpenSSLServerSocketFactory**  
*SSLServerSocketFactory that creates Server Sockets that use OpenSSL.*
- class **OpenSSLSocket**  
*Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.*

- class **OpenSSLSocketException**  
*Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.*
- class **OpenSSLSocketFactory**  
*Client Socket Factory that creates SSL based client sockets using the OpenSSL library.*
- class **OpenSSLSocketInputStream**  
*An output stream for reading data from an OpenSSL Socket instance.*
- class **OpenSSLSocketOutputStream**  
*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2808) instance.*

## 5.37 decaf::internal::net::tcp Namespace Reference

### Data Structures

- class **TcpSocket**  
*Platform-independent implementation of the socket interface.*
- class **TcpSocketInputStream**  
*Input stream for performing reads on a socket.*
- class **TcpSocketOutputStream**  
*Output stream for performing write operations on a socket.*

## 5.38 decaf::internal::nio Namespace Reference

### Data Structures

- class **BufferFactory**  
*Factory class used by static methods in the **decaf::nio** (p. 136) package to create the various default version of the NIO interfaces.*
- class **ByteBuffer**  
*This class defines six categories of operations upon byte buffers:*
  - class **CharArrayBuffer**
  - class **DoubleArrayBuffer**
  - class **FloatArrayBuffer**
  - class **IntArrayBuffer**
  - class **LongArrayBuffer**
  - class **ShortArrayBuffer**

## 5.39 decaf::internal::security Namespace Reference

### Data Structures

- class **SecureRandomImpl**

*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

## 5.40 decaf::internal::util Namespace Reference

### Namespaces

- namespace **concurrent**

### Data Structures

- class **ByteArrayAdapter**

*This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.*

- class **GenericResource**

*A Generic **Resource** (p. 3223) wraps some type and will delete it when the **Resource** (p. 3223) itself is deleted.*

- class **HexStringParser**

- class **Resource**

*Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.*

- class **ResourceLifecycleManager**

- class **TimerTaskHeap**

*A Binary Heap implemented specifically for the Timer class in Decaf Util.*

## 5.41 decaf::internal::util::concurrent Namespace Reference

### Data Structures

- class **ConditionImpl**

- class **MutexImpl**

- class **SynchronizableImpl**

*A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.*

- class **Transferer**

*Shared internal API for dual stacks and queues.*

- class **TransferQueue**

*This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.*

- class **TransferStack**

## 5.42 decaf::io Namespace Reference

### Data Structures

- class **BlockingByteArrayInputStream**

*This is a blocking version of a byte buffer stream.*

- class **BufferedInputStream**

*A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.*

- class **BufferedOutputStream**

*Wrapper around another output stream that buffers output before writing to the target output stream.*

- class **ByteArrayInputStream**

*A **ByteArrayInputStream** (p. 984) contains an internal buffer that contains bytes that may be read from the stream.*

- class **ByteArrayOutputStream**

- class **Closeable**

*Interface for a class that implements the close method.*

- class **DataInput**

*The **DataInput** (p. 1523) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

- class **DataInputStream**

*A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.*

- class **DataOutput**

*The **DataOutput** (p. 1541) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

- class **DataOutputStream**

*A data output stream lets an application write primitive Java data types to an output stream in a portable way.*

- class **EOFException**

- class **FileDescriptor**

*This class serves as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.*

- class **FilterInputStream**

*A **FilterInputStream** (p. 1854) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

- class **FilterOutputStream**

*This class is the superclass of all classes that filter output streams.*



- class **Flushable**  
*A **Flushable** (p. 1899) is a destination of data that can be flushed.*
- class **InputStream**  
*A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.*
- class **InputStreamReader**  
*An **InputStreamReader** (p. 2013) is a bridge from byte streams to character streams.*
- class **InterruptedIOException**
- class **IOException**
- class **OutputStream**  
*Base interface for any class that wants to represent an output stream of bytes.*
- class **OutputStreamWriter**  
*A class for turning a character stream into a byte stream.*
- class **PushbackInputStream**  
*A **PushbackInputStream** (p. 3086) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*
- class **Reader**
- class **UnsupportedEncodingException**  
*Thrown when the the Character Encoding is not supported.*
- class **UTFDataFormatException**  
*Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.*
- class **Writer**

## 5.43 decaf::lang Namespace Reference

### Namespaces

- namespace **exceptions**

### Data Structures

- class **Appendable**  
*An object to which char sequences and values can be appended.*
- class **ArrayPointer**  
*Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.*
- class **ArrayPointerComparator**  
*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 697).*
- class **Boolean**
- class **Byte**
- class **Character**

- class **CharSequence**

*A **CharSequence** (p. 1107) is a readable sequence of char values.*

- class **Comparable**

*This interface imposes a total ordering on the objects of each class that implements it.*

- class **Double**

- class **Exception**

- class **Float**

- class **Integer**

- class **Iterable**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 2112) type for generic collections API calls.*

- class **Long**

- class **Math**

*The class **Math** (p. 2455) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

*The abstract class **Number** (p. 2786) is the superclass of classes **Byte** (p. 918), **Double** (p. 1751), **Float** (p. 1865), **Integer** (p. 2038), **Long** (p. 2377), and **Short** (p. 3380).*

- struct **STATIC\_CAST\_TOKEN**

- struct **DYNAMIC\_CAST\_TOKEN**

- class **Pointer**

*Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.*

- class **PointerComparator**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2896) instance.*

- class **Readable**

*A **Readable** (p. 3106) is a source of characters.*

- class **Runnable**

*Interface for a runnable object - defines a task that can be run by a thread.*

- class **Runtime**

- class **Short**

- class **String**

*The **String** (p. 3610) class represents an immutable sequence of chars.*

- class **System**

*The **System** (p. 3670) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

- class **Thread**

*A **Thread** (p. 3707) is a concurrent unit of execution.*

- class **ThreadGroup**

- class **Throwable**

*This class represents an error that has occurred.*

## Functions

- `template<typename T , typename R , typename U >`  
`bool operator== (const ArrayPointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator== (const U *left, const ArrayPointer< T, R > &right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const ArrayPointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const U *left, const ArrayPointer< T, R > &right)`
- `template<typename T , typename R , typename U >`  
`bool operator== (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator== (const U *left, const Pointer< T, R > &right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool operator!= (const U *left, const Pointer< T, R > &right)`

### 5.43.1 Function Documentation

5.43.1.1 `template<typename T , typename R , typename U > bool decaf::lang::operator!= (`  
`const ArrayPointer< T, R > &left, const U * right ) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.43.1.2 `template<typename T , typename R , typename U > bool decaf::lang::operator!= (`  
`const U * left, const ArrayPointer< T, R > &right ) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.43.1.3 `template<typename T , typename R , typename U > bool decaf::lang::operator!= (`  
`const U * left, const Pointer< T, R > &right ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.4 `template<typename T , typename R , typename U > bool decaf::lang::operator!= (`  
`const Pointer< T, R > &left, const U * right ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.5 `template<typename T , typename R , typename U > bool decaf::lang::operator== (`  
`const Pointer< T, R > &left, const U * right ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.6 `template<typename T , typename R , typename U > bool decaf::lang::operator== ( const U * left, const ArrayPointer< T, R > & right ) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

5.43.1.7 `template<typename T , typename R , typename U > bool decaf::lang::operator== ( const U * left, const Pointer< T, R > & right ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.43.1.8 `template<typename T , typename R , typename U > bool decaf::lang::operator== ( const ArrayPointer< T, R > & left, const U * right ) [inline]`

References `decaf::lang::ArrayPointer< T, REFCOUNTER >::get()`.

## 5.44 decaf::lang::exceptions Namespace Reference

### Data Structures

- class **ClassCastException**
- class **IllegalArgumentException**
- class **IllegalMonitorStateException**
- class **IllegalStateException**
- class **IllegalThreadStateException**
- class **IndexOutOfBoundsException**
- class **InterruptedException**
- class **InvalidStateException**
- class **NoSuchElementException**
- class **NullPointerException**
- class **NumberFormatException**
- class **RuntimeException**
- class **UnsupportedOperationException**

## 5.45 decaf::net Namespace Reference

### Namespaces

- namespace **ssl**

## Data Structures

- class **BindException**
- class **ConnectException**
- class **HttpRetryException**
- class **Inet4Address**
- class **Inet6Address**
- class **InetAddress**
  - Represents an IP address.*
- class **InetSocketAddress**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**
  - This class implements server sockets.*
- class **ServerSocketFactory**
  - Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.*
- class **Socket**
- class **SocketAddress**
  - Base class for protocol specific **Socket** (p. 3445) addresses.*
- class **SocketError**
  - Static utility class to simplify handling of error codes for socket operations.*
- class **SocketException**
  - Exception for errors when manipulating sockets.*
- class **SocketFactory**
  - The **SocketFactory** (p. 3467) is used to create **Socket** (p. 3445) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*
- class **SocketImpl**
  - Acts as a base class for all physical **Socket** (p. 3445) implementations.*
- class **SocketImplFactory**
  - Factory class interface for a Factory that creates SocketImpl objects.*
- class **SocketOptions**
- class **SocketTimeoutException**
- class **UnknownHostException**
- class **UnknownServiceException**
- class **URI**
  - This class represents an instance of a **URI** (p. 3853) as defined by RFC 2396.*
- class **URISyntaxException**
- class **URL**
  - Class **URL** (p. 3891) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*
- class **URLDecoder**
- class **URLEncoder**

## 5.46 decaf::net::ssl Namespace Reference

### Data Structures

- class **SSLContext**  
*Represents an implementation of the Secure **Socket** (p. 3445) Layer for streaming based sockets.*
- class **SSLContextSpi**  
*Defines the interface that should be provided by an **SSLContext** (p. 3489) provider.*
- class **SSLParameters**
- class **SSLServerSocket**  
*Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.*
- class **SSLServerSocketFactory**  
*Factory class interface that provides methods to create SSL Server Sockets.*
- class **SSLSocket**
- class **SSLSocketFactory**  
*Factory class interface for a **SocketFactory** (p. 3467) that can create **SSLSocket** (p. 3506) objects.*

## 5.47 decaf::nio Namespace Reference

### Data Structures

- class **Buffer**  
*A container for data of a specific primitive type.*
- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**  
*This class defines six categories of operations upon byte buffers:*
- class **CharBuffer**  
*This class defines four categories of operations upon character buffers:*
- class **DoubleBuffer**  
*This class defines four categories of operations upon double buffers:*
- class **FloatBuffer**  
*This class defines four categories of operations upon float buffers:*
- class **IntBuffer**  
*This class defines four categories of operations upon int buffers:*
- class **InvalidMarkException**
- class **LongBuffer**  
*This class defines four categories of operations upon long long buffers:*
- class **ReadOnlyBufferException**
- class **ShortBuffer**  
*This class defines four categories of operations upon short buffers:*

## 5.48 decaf::security Namespace Reference

### Namespaces

- namespace **auth**
- namespace **cert**

### Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

*The **Key** (p. 2253) interface is the top-level interface for all keys.*

- class **KeyException**
- class **KeyManagementException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

*Base interface for a principal, which can represent an individual or organization.*

- class **PublicKey**

*A public key.*

- class **SecureRandom**
- class **SecureRandomSpi**

*Interface class used by Security Service Providers to implement a source of secure random bytes.*

- class **SignatureException**

## 5.49 decaf::security::auth Namespace Reference

### Namespaces

- namespace **x500**

## 5.50 decaf::security::auth::x500 Namespace Reference

### Data Structures

- class **X500Principal**

## 5.51 decaf::security::cert Namespace Reference

### Data Structures

- class **Certificate**  
*Base interface for all identity certificates.*
- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**
- class **X509Certificate**  
*Base interface for all identity certificates.*

## 5.52 decaf::util Namespace Reference

### Namespaces

- namespace **comparators**
- namespace **concurrent**
- namespace **logging**
- namespace **zip**

### Data Structures

- class **AbstractCollection**  
*This class provides a skeletal implementation of the **Collection** (p. 1155) interface, to minimize the effort required to implement this interface.*
- class **AbstractList**  
*This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*
- class **AbstractMap**  
*This class provides a skeletal implementation of the **Map** (p. 2419) interface, to minimize the effort required to implement this interface.*
- class **AbstractQueue**  
*This class provides skeletal implementations of some **Queue** (p. 3094) operations.*
- class **AbstractSequentialList**  
*This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*
- class **AbstractSet**  
*This class provides a skeletal implementation of the **Set** (p. 3379) interface to minimize the effort required to implement this interface.*



- class **Collection**  
*The root interface in the collection hierarchy.*
- class **Comparator**  
*A comparison function, which imposes a total ordering on some collection of objects.*
- class **Date**  
*Wrapper class around a time value in milliseconds.*
- class **Iterator**  
*Defines an object that can be used to iterate over the elements of a collection.*
- class **List**  
*An ordered collection (also known as a sequence).*
- class **ListIterator**  
*An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.*
- class **Map**  
***Map** (p. 2419) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **PriorityQueue**  
*An unbounded priority queue based on a binary heap algorithm.*
- class **Properties**  
*Java-like properties class for mapping string names to string values.*
- class **Queue**  
*A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.*
- class **Random**  
***Random** (p. 3100) Value Generator which is used to generate a stream of pseudorandom numbers.*
- class **Set**  
*A collection that contains no duplicate elements.*
- class **StlList**  
***List** (p. 2296) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*
- class **StlMap**  
***Map** (p. 2419) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **StlQueue**  
*The **Queue** (p. 3094) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*
- class **StlSet**  
***Set** (p. 3379) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.*
- class **StringTokenizer**
- class **Timer**  
*A facility for threads to schedule tasks for future execution in a background thread.*
- class **TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3730).*

- class **UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3900)).*

## 5.53 decaf::util::comparators Namespace Reference

### Data Structures

- class **Less**

*Simple **Less** (p. 2287) **Comparator** (p. 1189) that compares to elements to determine if the first is less than the second.*

## 5.54 decaf::util::concurrent Namespace Reference

### Namespaces

- namespace **atomic**
- namespace **locks**

### Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **BlockingQueue**

*A **decaf::util::Queue** (p. 3094) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

- class **BrokenBarrierException**
- class **Callable**

*A task that returns a result and may throw an exception.*

- class **CancellationException**
- class **ConcurrentMap**

*Interface for a **Map** (p. 2419) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2419) interface.*

- class **ConcurrentStlMap**

***Map** (p. 2419) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **CountDownLatch**
- class **Delayed**

*A mix-in style interface for marking objects that should be acted upon after a given delay.*

- class **ExecutionException**

- class **Executor**  
An object that executes submitted **decaf.lang Runnable** (p. 3264) tasks.
- class **ExecutorService**  
An **Executor** (p. 1831) that provides methods to manage termination and methods that can produce a **Future** (p. 1929) for tracking progress of one or more asynchronous tasks.
- class **Future**  
A **Future** (p. 1929) represents the result of an asynchronous computation.
- class **Lock**  
A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.
- class **Mutex**  
**Mutex** (p. 2736) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.
- class **PooledThread**
- class **PooledThreadListener**  
Abstract Listener Interface for users of **ThreadPool** (p. 3718) .
- class **RejectedExecutionException**
- class **RejectedExecutionHandler**  
A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).
- class **Semaphore**  
A counting semaphore.
- class **Synchronizable**  
The interface for all synchronizable objects (that is, objects that can be locked and unlocked).
- class **SynchronousQueue**  
A **blocking queue** (p. 804) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.
- class **TaskListener**
- class **ThreadFactory**  
public interface **ThreadFactory** (p. 3716)
- class **ThreadPool**  
Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.
- class **TimeoutException**
- class **TimeUnit**  
A **TimeUnit** (p. 3748) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

## 5.55 decaf::util::concurrent::atomic Namespace Reference

### Data Structures

- class **AtomicBoolean**

*A boolean value that may be updated atomically.*

- class **AtomicInteger**

*An int value that may be updated atomically.*

- class **AtomicRefCounter**
- class **AtomicReference**

*An Pointer reference that may be updated atomically.*

## 5.56 decaf::util::concurrent::locks Namespace Reference

### Data Structures

- class **Condition**

**Condition** (p. 1220) factors out the **Mutex** (p. 2736) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2336) implementations.

- class **Lock**

**Lock** (p. 2336) implementations provide more extensive locking operations than can be obtained using synchronized statements.

- class **LockSupport**

*Basic thread blocking primitives for creating locks and other synchronization classes.*

- class **ReadWriteLock**

A **ReadWriteLock** (p. 3117) maintains a pair of associated locks, one for read-only operations and one for writing.

- class **ReentrantLock**

A reentrant mutual exclusion **Lock** (p. 2336) with extended capabilities.

## 5.57 decaf::util::logging Namespace Reference

### Data Structures

- class **ConsoleHandler**

This **Handler** (p. 1941) publishes log records to System.err.

- class **ErrorManager**

**ErrorManager** (p. 1792) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1941) during Logging.

- class **Filter**

A **Filter** (p. 1853) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

- class **Formatter**

A **Formatter** (p. 1927) provides support for formatting LogRecords.

- class **Handler**

A **Handler** (p. 1941) object takes log messages from a **Logger** (p. 2345) and exports them.

- class **Level**  
*The **Level** (p. 2290) class defines a set of standard logging levels that can be used to control logging output.*
- class **Logger**  
*A **Logger** (p. 2345) object is used to log messages for a specific system or application component.*
- class **LoggerHierarchy**
- class **LogManager**  
*There is a single global **LogManager** (p. 2363) object that is used to maintain a set of shared state about Loggers and log services.*
- class **LogRecord**  
***LogRecord** (p. 2370) objects are used to pass logging requests between the logging framework and individual log Handlers.*
- class **LogWriter**
- class **MarkBlockLogger**  
*Defines a class that can be used to mark the entry and exit from scoped blocks.*
- class **PropertiesChangeListener**  
*Defines the interface that classes can use to listen for change events on **Properties** (p. 3072).*
- class **SimpleFormatter**  
*Print a brief summary of the **LogRecord** (p. 2370) in a human readable format.*
- class **SimpleLogger**
- class **StreamHandler**  
*Stream based logging **Handler** (p. 1941).*
- class **XMLFormatter**  
*Format a **LogRecord** (p. 2370) into a standard XML format.*

## Enumerations

- enum **Levels** {  
    **Off**, **Null**, **Markblock**, **Debug**,  
    **Info**, **Warn**, **Error**, **Fatal**,  
    **Throwing** }  
*Defines an enumeration for logging levels.*

### 5.57.1 Enumeration Type Documentation

#### 5.57.1.1 enum decaf::util::logging::Levels

Defines an enumeration for logging levels.

#### Enumerator:

**Off**

**Null**

**Markblock**

**Debug**

**Info**

**Warn**

**Error**

**Fatal**

**Throwing**

## 5.58 decaf::util::zip Namespace Reference

### Data Structures

- class **Adler32**  
*Clas that can be used to compute an Adler-32 **Checksum** (p. 1114) for a data stream.*
- class **CheckedInputStream**  
*An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1114) of the bytes read, the **Checksum** (p. 1114) can then be used to verify the integrity of the input stream.*
- class **CheckedOutputStream**  
*An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 1114) of the bytes written, the **Checksum** (p. 1114) can then be used to verify the integrity of the output stream.*
- class **Checksum**  
*An interface used to represent **Checksum** (p. 1114) values in the Zip package.*
- class **CRC32**  
*Class that can be used to compute a CRC-32 checksum for a data stream.*
- class **DataFormatException**
- class **Deflater**  
*This class compresses data using the DEFLATE algorithm (see `specification`).*
- class **DeflaterOutputStream**  
*Provides a `FilterOutputStream` instance that compresses the data before writing it to the wrapped `OutputStream`.*
- class **Inflater**  
*This class uncompresses data that was compressed using the DEFLATE algorithm (see `specification`).*
- class **InflaterInputStream**  
*A `FilterInputStream` that decompresses data read from the wrapped `InputStream` instance.*
- class **ZipException**

## 5.59 std Namespace Reference

### Data Structures

- struct **less**< **decaf::lang::ArrayPointer**< **T** > >

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

- struct **less**< **decaf::lang::Pointer**< **T** > >

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*





## Chapter 6

# Data Structure Documentation

### 6.1 decaf::util::AbstractCollection< E > Class Template Reference

This class provides a skeletal implementation of the **Collection** (p. 1155) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractCollection.h>
```

Inheritance diagram for decaf::util::AbstractCollection< E >:

#### Public Member Functions

- **AbstractCollection** ()
- virtual ~**AbstractCollection** ()
- **AbstractCollection**< E > & **operator=** (const **AbstractCollection**< E > &collection)  
*Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.*
- virtual bool **add** (const E &value DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Ensures that this collection contains the specified element (optional operation).*
- virtual bool **addAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all of the elements in the specified collection to this collection (optional operation).*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all of the elements from this collection (optional operation).*
- virtual void **copy** (const **Collection**< E > &collection)

*Renders this **Collection** (p. 1155) as a Copy of the given **Collection** (p. 1155).*

- virtual bool **contains** (const E &value) const throw ( lang::Exception )

*Returns true if this collection contains the specified element.*

- virtual bool **containsAll** (const **Collection**< E > &collection) const throw ( lang::Exception )

*Returns true if this collection contains all of the elements in the specified collection.*

- virtual bool **equals** (const **Collection**< E > &collection) const

*Answers true if this **Collection** (p. 1155) and the one given are the same size and if each element contained in the **Collection** (p. 1155) given is equal to an element contained in this collection.*

- virtual bool **isEmpty** () const

*Returns true if this collection contains no elements.*

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

- virtual bool **removeAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes all of this collection's elements that are also contained in the specified collection (optional operation).*

- virtual bool **retainAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Retains only the elements in this collection that are contained in the specified collection (optional operation).*

- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1155).*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

## Protected Attributes

- **util::concurrent::Mutex mutex**

### 6.1.1 Detailed Description

```
template<typename E>class decaf::util::AbstractCollection< E >
```

This class provides a skeletal implementation of the **Collection** (p. 1155) interface, to minimize the effort required to implement this interface.

To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p. 1155) constructor, as per the recommendation in the **Collection** (p. 1155) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

## Since

1.0

### 6.1.2 Constructor & Destructor Documentation

```
6.1.2.1 template<typename E> decaf::util::AbstractCollection< E
>::AbstractCollection ( ) [inline]
```

```
6.1.2.2 template<typename E> virtual decaf::util::AbstractCollection< E
>::~~AbstractCollection ( ) [inline, virtual]
```

### 6.1.3 Member Function Documentation

```
6.1.3.1 template<typename E> virtual bool decaf::util::AbstractCollection<
    E >::add ( const E &value DECAF_UNUSED ) throw (
        lang::exceptions::UnsupportedOperationException,
        lang::exceptions::IllegalArgumentException,
        lang::exceptions::IllegalStateException ) [inline, virtual]
```

Ensures that this collection contains the specified element (optional operation).

Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. **Collection** (p. 1155) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an UnsupportedOperationException.

#### Parameters

<i>value</i>	- The element that must be ensured to be in this collection.
--------------	--

#### Returns

true if the collection was changed as a result of this call.

#### Exceptions

<i>UnsupportedOperationException</i>	if the add operation is not supported by this collection
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Referenced by decaf::util::AbstractCollection< cms::Connection \* >::addAll(), decaf::util::AbstractCollection< cms::Connection \* >::copy(), and decaf::util::AbstractCollection< cms::Connection \* >::operator=().

```
6.1.3.2 template<typename E> virtual bool decaf::util::AbstractCollection<
    E >::addAll ( const Collection< E > &collection ) throw
    ( lang::exceptions::UnsupportedOperationException,
      lang::exceptions::IllegalArgumentException,
      lang::exceptions::IllegalStateException ) [inline, virtual]
```

Adds all of the elements in the specified collection to this collection (optional operation).

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if

the specified collection is this collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1155) whose elements are to be added to this <b>Collection</b> (p. 1155).
-------------------	---

#### Returns

true if the collection was changed as a result of this call.

#### Exceptions

<i>UnsupportedOperationException</i>	if the addAll operation is not supported by this collection
<i>IllegalArgumentException</i>	if some property of the element prevents it from being added to this collection
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection**< E > (p. 1158).

Reimplemented in **decaf::util::AbstractQueue**< E > (p. 165).

```
6.1.3.3  template<typename E> virtual void decaf::util::AbstractCollection< E >::clear (
        ) throw ( lang::exceptions::UnsupportedOperationException ) [inline,
        virtual]
```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2115) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

#### Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Implements **decaf::util::Collection**< E > (p. 1158).

Reimplemented in **decaf::util::AbstractQueue**< E > (p. 166), **decaf::util::concurrent::SynchronousQueue**<

**E** > (p. 3662), **decaf::util::PriorityQueue< E >** (p. 2980), **decaf::util::StlList< E >** (p. 3537), **decaf::util::StlSet< E >** (p. 3568), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3537), **decaf::util::StlList< CompositeTask \* >** (p. 3537), **decaf::util::StlList< URI >** (p. 3537), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3537), **decaf::util::StlList< PrimitiveValueNode >** (p. 3537), **decaf::util::StlList< Pointer< Command > >** (p. 3537), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3537), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3537), **decaf::util::StlList< cms::Destination \* >** (p. 3537), **decaf::util::StlList< cms::Session \* >** (p. 3537), **decaf::util::StlList< cms::Connection \* >** (p. 3537), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3568), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3568), **decaf::util::StlSet< Resource \* >** (p. 3568), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3568).

Referenced by **decaf::util::AbstractCollection< cms::Connection \* >::copy()**, and **decaf::util::AbstractCollection< cms::Connection \* >::operator=()**.

```
6.1.3.4 template<typename E> virtual bool decaf::util::AbstractCollection< E
    >::contains ( const E & value ) const throw ( lang::Exception ) [inline,
    virtual]
```

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

#### Parameters

<i>value</i>	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1155).
--------------	--

#### Returns

true if the value is contained in this collection

#### Exceptions

<i>Exception</i>	if an error occurs,
------------------	---------------------

Implements **decaf::util::Collection< E >** (p. 1159).

Reimplemented in **decaf::util::StlList< E >** (p. 3537), **decaf::util::StlSet< E >** (p. 3569), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3537), **decaf::util::StlList< CompositeTask \* >** (p. 3537), **decaf::util::StlList< URI >** (p. 3537), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3537), **decaf::util::StlList< PrimitiveValueNode >** (p. 3537), **decaf::util::StlList< Pointer< Command > >** (p. 3537), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3537), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3537), **decaf::util::StlList< cms::Destination \* >** (p. 3537), **decaf::util::StlList< cms::Session \* >** (p. 3537), **decaf::util::StlList< cms::Connection \* >** (p. 3537), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3569), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3569), **decaf::util::StlSet< Resource \* >** (p. 3569), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3569).

Referenced by **decaf::util::AbstractCollection< cms::Connection \* >::containsAll()**.

```
6.1.3.5 template<typename E> virtual bool decaf::util::AbstractCollection<
    E>::containsAll ( const Collection< E > & collection ) const throw (
    lang::Exception ) [inline, virtual]
```

Returns true if this collection contains all of the elements in the specified collection.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

#### Parameters

<i>collection</i>	collection to be checked for containment in this collection
-------------------	---

#### Returns

true if this collection contains all of the elements in the specified collection.

#### Exceptions

<i>Exception</i>	if an error occurs,
------------------	---------------------

Implements **decaf::util::Collection< E >** (p. 1160).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3663).

Referenced by decaf::util::AbstractCollection< cms::Connection \* >::equals().

```
6.1.3.6 template<typename E> virtual void decaf::util::AbstractCollection< E >::copy (
    const Collection< E > & collection ) [inline, virtual]
```

Renders this **Collection** (p. 1155) as a Copy of the given **Collection** (p. 1155).

This implementation iterates over the contents of the given collection adding each to this collection after first calling this Collection's clear method.

#### Parameters

<i>collection</i>	- the collection to mirror.
-------------------	-----------------------------

```
6.1.3.7 template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals
    ( const Collection< E > & collection ) const [inline, virtual]
```

Answers true if this **Collection** (p. 1155) and the one given are the same size and if each element contained in the **Collection** (p. 1155) given is equal to an element contained in this collection.

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1155) to be compared to this one.
-------------------	---

**Returns**

true if this **Collection** (p. 1155) is equal to the one given.

Implements **decaf::util::Collection**< **E** > (p. 1161).

Reimplemented in **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3665).

**6.1.3.8** `template<typename E> virtual bool decaf::util::AbstractCollection< E  
>::isEmpty ( ) const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 1164) == 0.

**Returns**

true if the size method return 0.

Implements **decaf::util::Collection**< **E** > (p. 1161).

Reimplemented in **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3665), **decaf::util::StlList**< **E** > (p. 3539), **decaf::util::StlSet**< **E** > (p. 3570), **decaf::util::StlList**< **cms::MessageConsumer** \* > (p. 3539), **decaf::util::StlList**< **CompositeTask** \* > (p. 3539), **decaf::util::StlList**< **URI** > (p. 3539), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3539), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3539), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3539), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3539), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3539), **decaf::util::StlList**< **cms::Destination** \* > (p. 3539), **decaf::util::StlList**< **cms::Session** \* > (p. 3539), **decaf::util::StlList**< **cms::Connection** \* > (p. 3539), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3570), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3570), **decaf::util::StlSet**< **Resource** \* > (p. 3570), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3570).

Referenced by **decaf::util::AbstractQueue**< **E** >::clear().

**6.1.3.9** `template<typename E> virtual void decaf::util::AbstractCollection< E >::lock  
( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline,  
virtual]`

Locks the object.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).



```
6.1.3.10  template<typename E> virtual void decaf::util::AbstractCollection<
          E >::notify ( ) throw ( decaf::lang::exceptions::RuntimeException,
          decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
          virtual]
```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

```
6.1.3.11  template<typename E> virtual void decaf::util::AbstractCollection< E
          >::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException,
          decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
          virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

```
6.1.3.12  template<typename E> AbstractCollection<E>&
          decaf::util::AbstractCollection< E >::operator= ( const
          AbstractCollection< E > & collection ) [inline]
```

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

#### Parameters

<i>collection</i>	- the collection to copy
-------------------	--------------------------

#### Returns

a reference to this collection

```

6.1.3.13 template<typename E> virtual bool decaf::util::AbstractCollection<
    E >::remove ( const E & value ) throw (
        lang::exceptions::UnsupportedOperationException,
        lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

#### Parameters

<i>value</i>	- element to be removed from this collection, if present
--------------	--

#### Returns

true if an element was removed as a result of this call

#### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1155).

Implements **decaf::util::Collection< E >** (p. 1162).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2983), **decaf::util::StlList< E >** (p. 3541), **decaf::util::StlSet< E >** (p. 3570), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3541), **decaf::util::StlList< CompositeTask \* >** (p. 3541), **decaf::util::StlList< URI >** (p. 3541), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3541), **decaf::util::StlList< PrimitiveValueNode >** (p. 3541), **decaf::util::StlList< Pointer< Command > >** (p. 3541), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3541), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3541), **decaf::util::StlList< cms::Destination \* >** (p. 3541), **decaf::util::StlList< cms::Session \* >** (p. 3541), **decaf::util::StlList< cms::Connection \* >** (p. 3541), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3570), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3570), **decaf::util::StlSet< Resource \* >** (p. 3570), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3570).

```
6.1.3.14  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::removeAll ( const Collection< E > & collection ) throw
           ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes all of this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

#### Parameters

<i>collection</i>	- collection containing elements to be removed from this collection
-------------------	---

#### Returns

true if this collection changed as a result of the call

#### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection
<i>IllegalArgumentException.</i>	

Implements **decaf::util::Collection< E >** (p. 1163).

Reimplemented in **decaf::util::AbstractSet< E >** (p. 169), **decaf::util::AbstractSet< transport::TransportListener \* >** (p. 169), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 169), **decaf::util::AbstractSet< Resource \* >** (p. 169), and **decaf::util::AbstractSet< ActiveMQSession \* >** (p. 169).

```
6.1.3.15  template<typename E> virtual bool decaf::util::AbstractCollection<
           E >::retainAll ( const Collection< E > & collection ) throw
           ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

This implementation iterates over this collection, checking each element returned by the

iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

#### Parameters

<i>collection</i>	- collection containing elements to be retained in this collection
-------------------	--

#### Returns

true if this collection changed as a result of the call

#### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection
<i>IllegalArgumentException</i> .	

Implements **decaf::util::Collection< E >** (p. 1163).

6.1.3.16 `template<typename E> virtual std::vector<E> decaf::util::AbstractCollection< E>::toArray ( ) const [inline, virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1155).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

#### Returns

an vector of copies of all the elements from this **Collection** (p. 1155)

Implements **decaf::util::Collection< E >** (p. 1165).

Reimplemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3670).

6.1.3.17 `template<typename E> virtual bool decaf::util::AbstractCollection< E>::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

**6.1.3.18** `template<typename E> virtual void decaf::util::AbstractCollection< E >::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException )`  
`[inline, virtual]`

Unlocks the object.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

**6.1.3.19** `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )` `[inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

**6.1.3.20** `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )` `[inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add

a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

```
6.1.3.21  template<typename E> virtual void decaf::util::AbstractCollection< E >::wait (
            long long milliseconds ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline,
            virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or WAIT_INFINITE
---------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

### 6.1.4 Field Documentation

6.1.4.1 `template<typename E> util::concurrent::Mutex`  
`decaf::util::AbstractCollection< E >::mutex` [mutable, protected]

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::lock()`, `decaf::util::AbstractCollection< cms::Connection * >::notify()`, `decaf::util::AbstractCollection< cms::Connection * >::notifyAll()`, `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`, `decaf::util::AbstractCollection< cms::Connection * >::unlock()`, and `decaf::util::AbstractCollection< cms::Connection * >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractCollection.h`

## 6.2 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

```
#include <src/main/decaf/util/AbstractList.h>
```

Inheritance diagram for `decaf::util::AbstractList< E >`:

### Public Member Functions

- `virtual ~AbstractList ()`

#### 6.2.1 Detailed Description

```
template<typename E> class decaf::util::AbstractList< E >
```

This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

For sequential access data (such as a linked list), **AbstractSequentialList** (p. 167) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the `get(int)` and `size()` (p. 1164) methods.

To implement a modifiable list, the programmer must additionally override the `set(int, E)` method (which otherwise throws an `UnsupportedOperationException`). If the list is variable-size the programmer must additionally override the `add(int, E)` and `remove(int)` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 1155) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: `get(int)`, `set(int, E)`, `add(int, E)` and `remove(int)`.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

#### Since

1.0

### 6.2.2 Constructor & Destructor Documentation

6.2.2.1 `template<typename E> virtual decaf::util::AbstractList< E >::~~AbstractList ( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractList.h`

### 6.3 `decaf::util::AbstractMap< K, V, COMPARATOR >` Class Template Reference

This class provides a skeletal implementation of the **Map** (p. 2419) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractMap.h>
```

Inheritance diagram for `decaf::util::AbstractMap< K, V, COMPARATOR >`:

#### Public Member Functions

- `virtual ~AbstractMap ()`

#### 6.3.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR>class decaf::util::AbstractMap< K, V, COMPARATOR >
```

This class provides a skeletal implementation of the **Map** (p. 2419) interface, to minimize the effort required to implement this interface.



To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the `entrySet` method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p. 168). This set should not support the `add` or `remove` methods, and its iterator should not support the `remove` method.

To implement a modifiable map, the programmer must additionally override this class's `put` method (which otherwise throws an `UnsupportedOperationException`), and the iterator returned by `entrySet().iterator()` must additionally implement its `remove` method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p. 2419) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

#### Since

1.0

### 6.3.2 Constructor & Destructor Documentation

```
6.3.2.1 template<typename K , typename V , typename COMPARATOR > virtual
decaf::util::AbstractMap< K, V, COMPARATOR >::~~AbstractMap ( )
[inline, virtual]
```

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractMap.h`

## 6.4 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 3094) operations.

```
#include <src/main/decaf/util/AbstractQueue.h>
```

Inheritance diagram for `decaf::util::AbstractQueue< E >`:

### Public Member Functions

- **AbstractQueue** ()
- virtual **~AbstractQueue** ()
- virtual bool **add** (const E &value) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.*

- virtual bool **addAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

*Adds all the elements of a collection to the queue.*

- virtual E **remove** () throw ( decaf::lang::exceptions::NoSuchElementException )

*Retrieves and removes the head of this queue.*

- virtual E **element** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Retrieves, but does not remove, the head of this queue.*

- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )

*Removes all elements of the queue.*

### 6.4.1 Detailed Description

```
template<typename E>class decaf::util::AbstractQueue< E >
```

This class provides skeletal implementations of some **Queue** (p. 3094) operations.

Methods add, remove, and element are based on offer, poll, and peek, respectively.

A **Queue** (p. 3094) implementation that extends this class must minimally define a method **Queue** (p. 3094). **offer(E)** which does not permit insertion of null elements, along with methods **Queue** (p. 3094). **peek()** (p. 3097), **Queue.poll()** (p. 3097), **Collection.size()** (p. 1164), and a **Collection.iterator()** (p. 2113) supporting **Iterator.remove()** (p. 2115). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 147).

**Since**

1.0

### 6.4.2 Constructor & Destructor Documentation

```
6.4.2.1 template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue ( )
[inline]
```

```
6.4.2.2 template<typename E > virtual decaf::util::AbstractQueue< E
>::~AbstractQueue ( ) [inline, virtual]
```

### 6.4.3 Member Function Documentation

```
6.4.3.1 template<typename E > virtual bool decaf::util::AbstractQueue<
    E >::add ( const E & value ) throw ( de-
    caf::lang::exceptions::UnsupportedOperationException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalStateException ) [inline, virtual]
```

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

#### Parameters

<i>value</i>	- the element to offer to the <b>Queue</b> (p. 3094).
--------------	---

#### Returns

true if the add succeeds.

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if the element cannot be added.
---------------------------------------	---------------------------------

Implements **decaf::util::Collection< E >** (p. 1156).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2979).

References **decaf::util::Queue< E >::offer()**.

```
6.4.3.2 template<typename E > virtual bool decaf::util::AbstractQueue<
    E >::addAll ( const Collection< E > & collection ) throw
    ( lang::exceptions::UnsupportedOperationException,
    lang::exceptions::IllegalArgumentException,
    lang::exceptions::IllegalStateException ) [inline, virtual]
```

Adds all the elements of a collection to the queue.

If the collection is the queue itself, then an `IllegalArgumentException` will be thrown out. If during the process, some runtime exception is thrown out, then part of the elements in the collection that have successfully added will remain in the queue.

The result of the method is undefined if the collection is modified during the process of the method.

#### Parameters

<i>collection</i>	- the collection to be added to the queue.
-------------------	--

#### Returns

true if the operation succeeds.

**Exceptions**

<i>IllegalArgumentException</i>	If the collection to be added to the queue is the queue itself.
---------------------------------	---

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 150).

```
6.4.3.3  template<typename E> virtual void decaf::util::AbstractQueue< E >::clear ( )
        throw ( lang::exceptions::UnsupportedOperationException ) [inline,
        virtual]
```

Removes all elements of the queue.

This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 151).

Reimplemented in **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3662), and **decaf::util::PriorityQueue**< **E** > (p. 2980).

References decaf::util::AbstractCollection< **E** >::isEmpty(), and decaf::util::Queue< **E** >::poll().

```
6.4.3.4  template<typename E> virtual E decaf::util::AbstractQueue< E >::element
        ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )
        [inline, virtual]
```

Retrieves, but does not remove, the head of this queue.

This method differs from peek only in that it throws an exception if this queue is empty.

This implementation returns the result of peek unless the queue is empty.

**Returns**

the element in the head of the queue.

**Exceptions**

<i>NoSuchElementException</i>	if the queue is empty.
-------------------------------	------------------------

Implements **decaf::util::Queue**< **E** > (p. 3096).

References decaf::util::Queue< **E** >::peek().

Referenced by decaf::util::concurrent::SynchronousQueue< **E** >::drainTo().

```
6.4.3.5 template<typename E> virtual E decaf::util::AbstractQueue< E >::remove ( )
        throw ( decaf::lang::exceptions::NoSuchElementException ) [inline,
        virtual]
```

Retrieves and removes the head of this queue.

This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

### Returns

a copy of the element in the head of the queue.

### Exceptions

<i>NoSuchElementException</i>	if the queue is empty.
-------------------------------	------------------------

Implements **decaf::util::Queue< E >** (p. 3098).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2982).

References **decaf::util::Queue< E >::poll()**.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractQueue.h**

## 6.5 decaf::util::AbstractSequentialList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

```
#include <src/main/decaf/util/AbstractSequentialList.h>
```

Inheritance diagram for **decaf::util::AbstractSequentialList< E >**:

### Public Member Functions

- virtual **~AbstractSequentialList** ()

### 6.5.1 Detailed Description

```
template<typename E>class decaf::util::AbstractSequentialList< E >
```

This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

For random access data (such as an array), **AbstractList** (p. 161) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p. 161) class in the sense that it implements the "random access" methods (`get(int index)`, `set(int index, E element)`, `add(int index, E element)` and `remove(int index)`) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the `listIterator` and `size` methods. For an unmodifiable list, the programmer need only implement the list iterator's `hasNext`, `next`, `hasPrevious`, `previous` and `index` methods.

For a modifiable list the programmer should additionally implement the list iterator's `set` method. For a variable-size list the programmer should additionally implement the list iterator's `remove` and `add` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 1155) interface specification.

Since

1.0

## 6.5.2 Constructor & Destructor Documentation

6.5.2.1 `template<typename E > virtual decaf::util::AbstractSequentialList< E >::~~AbstractSequentialList( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSequentialList.h`

## 6.6 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 3379) interface to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractSet.h>
```

Inheritance diagram for `decaf::util::AbstractSet< E >`:

## Public Member Functions

- virtual `~AbstractSet()`
- virtual bool **removeAll** (const **Collection**< E > &collection) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes from this set all of its elements that are contained in the specified collection (optional operation).*

### 6.6.1 Detailed Description

```
template<typename E>class decaf::util::AbstractSet< E >
```

This class provides a skeletal implementation of the **Set** (p. 3379) interface to minimize the effort required to implement this interface.

The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p. 1155) by extending **AbstractCollection** (p. 147), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 3379) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since

1.0

### 6.6.2 Constructor & Destructor Documentation

6.6.2.1 `template<typename E> virtual decaf::util::AbstractSet< E >::~~AbstractSet ( ) [inline, virtual]`

### 6.6.3 Member Function Documentation

6.6.3.1 `template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll ( const Collection< E > & collection ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [inline, virtual]`

Removes from this set all of its elements that are contained in the specified collection (optional operation).

If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer

elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method.

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1155) whose elements are to be retained
-------------------	---

#### Returns

true if the collection changed as a result of this call

#### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 157).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractSet.h**

## 6.7 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the **TransportFactory** (p. 3825) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3825) instances.

```
#include <src/main/activemq/transport/AbstractTransportFactory.h>
```

Inheritance diagram for `activemq::transport::AbstractTransportFactory`:

#### Public Member Functions

- virtual **~AbstractTransportFactory** ()

#### Protected Member Functions

- virtual **Pointer**< **wireformat::WireFormat** > **createWireFormat** (const **decaf::util::Properties** &properties) throw ( **decaf::lang::exceptions::NoSuchElementException** )

*Creates the WireFormat that is configured for this **Transport** (p. 3819) and returns it.*



### 6.7.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 3825) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3825) instances.

#### Since

3.0

### 6.7.2 Constructor & Destructor Documentation

6.7.2.1 `virtual activemq::transport::AbstractTransportFactory::~~AbstractTransportFactory ( )`  
`[inline, virtual]`

### 6.7.3 Member Function Documentation

6.7.3.1 `virtual Pointer<wireformat::WireFormat>`  
`activemq::transport::AbstractTransportFactory::createWireFormat`  
`( const decaf::util::Properties & properties ) throw (`  
`decaf::lang::exceptions::NoSuchElementException ) [protected,`  
`virtual]`

Creates the WireFormat that is configured for this **Transport** (p. 3819) and returns it.

The default WireFormat is Openwire.

#### Parameters

<i>properties</i>	The properties that were configured on the URI.
-------------------	---

#### Returns

a pointer to a WireFormat instance that the caller then owns.

#### Exceptions

<i>NoSuchElementException</i>	if the configured WireFormat is not found.
-------------------------------	--

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/AbstractTransportFactory.h`

## 6.8 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

```
#include <src/main/activemq/core/ActiveMQAckHandler.h>
```

## Public Member Functions

- virtual **~ActiveMQAckHandler** ()
- virtual void **acknowledgeMessage** (const **commands::Message** \*message)=0  
throw ( cms::CMSException )  
*Method called to acknowledge the message passed.*

### 6.8.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

#### Since

2.0

### 6.8.2 Constructor & Destructor Documentation

- 6.8.2.1 virtual **activemq::core::ActiveMQAckHandler::~ActiveMQAckHandler** ( )  
[inline, virtual]

### 6.8.3 Member Function Documentation

- 6.8.3.1 virtual void **activemq::core::ActiveMQAckHandler::acknowledgeMessage** ( const **commands::Message** \* *message* ) throw ( **cms::CMSException** ) [pure virtual]

Method called to acknowledge the message passed.

#### Parameters

<i>message</i>	Message to Acknowledge
----------------	------------------------

#### Exceptions

<i>CMSException</i>
---------------------

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQAckHandler.h**

## 6.9 activemq::commands::ActiveMQBlobMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQBlobMessage.h>
```

Inheritance diagram for **activemq::commands::ActiveMQBlobMessage**:

## Public Member Functions

- **ActiveMQBlobMessage** ()
- virtual **~ActiveMQBlobMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQBlobMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- std::string **getRemoteBlobUrl** () const  
*Get the Remote URL of the Blob.*
- void **setRemoteBlobUrl** (const std::string &remoteURL)  
*Set the Remote URL of the Blob.*
- std::string **getMimeType** () const  
*Get the Mime Type of the Blob.*
- void **setMimeType** (const std::string &mimeType)  
*Set the Mime Type of the Blob.*
- std::string **getName** () const  
*Gets the Name of the Blob.*
- void **setName** (const std::string &name)  
*Sets the Name of the Blob.*
- bool **isDeletedByBroker** () const  
*Gets if this Blob is deleted by the Broker.*
- void **setDeletedByBroker** (bool value)  
*Sets the Deleted By Broker flag.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY\_MIME\_TYPE**

### 6.9.1 Constructor & Destructor Documentation

6.9.1.1 `activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ( )`

6.9.1.2 `virtual activemq::commands::ActiveMQBlobMessage::~~ActiveMQBlobMessage ( )`  
`[inline, virtual]`

### 6.9.2 Member Function Documentation

6.9.2.1 `virtual cms::Message* activemq::commands::ActiveMQBlobMessage::clone ( )`  
`const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements `cms::Message` (p.2498).

6.9.2.2 `virtual ActiveMQBlobMessage* activemq::commands::ActiveMQBlobMessage::cloneDataStructure ( ) const` `[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p.2480).

6.9.2.3 `virtual void activemq::commands::ActiveMQBlobMessage::copyDataStructure ( const DataStructure * src )` `[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p.2481).

6.9.2.4 `virtual bool activemq::commands::ActiveMQBlobMessage::equals ( const  
DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 399).

6.9.2.5 `virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType  
( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Message** (p. 2483).

6.9.2.6 `std::string activemq::commands::ActiveMQBlobMessage::getMimeType ( ) const  
[inline]`

Get the Mime Type of the Blob.

#### Returns

string holding the MIME Type.

6.9.2.7 `std::string activemq::commands::ActiveMQBlobMessage::getName ( ) const  
[inline]`

Gets the Name of the Blob.

#### Returns

string name of the Blob.

6.9.2.8 `std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl ( ) const`  
[inline]

Get the Remote URL of the Blob.

#### Returns

string from of the Remote Blob URL.

6.9.2.9 `bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker ( ) const`  
[inline]

Gets if this Blob is deleted by the Broker.

#### Returns

true if the Blob is deleted by the Broker.

6.9.2.10 `void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker ( bool value )`  
[inline]

Sets the Deleted By Broker flag.

#### Parameters

<i>value</i>	- set the Delete by broker flag to value.
--------------	---

6.9.2.11 `void activemq::commands::ActiveMQBlobMessage::setMimeType ( const std::string & mimeType )` [inline]

Set the Mime Type of the Blob.

#### Parameters

<i>mimeType</i>	- String holding the MIME Type.
-----------------	---------------------------------

6.9.2.12 `void activemq::commands::ActiveMQBlobMessage::setName ( const std::string & name )` [inline]

Sets the Name of the Blob.

#### Parameters

<i>name</i>	- Name of the Blob.
-------------	---------------------

## 6.10

### activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference 177

6.9.2.13 `void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl ( const  
std::string & remoteURL ) [inline]`

Set the Remote URL of the Blob.

#### Parameters

<i>remoteURL</i>	- String form of the Remote URL.
------------------	----------------------------------

6.9.2.14 `virtual std::string activemq::commands::ActiveMQBlobMessage::toString ( ) const  
[virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2490).

## 6.9.3 Field Documentation

6.9.3.1 `const std::string activemq::commands::ActiveMQBlobMessage::BINARY_  
MIME_TYPE [static]`

6.9.3.2 `const unsigned char activemq::commands::ActiveMQBlobMessage::ID_  
ACTIVEMQBLOBMESSAGE = 29 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQBlobMessage.h**

## 6.10 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 177).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**:

## Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.10.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 177).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.10.2 Constructor & Destructor Documentation

6.10.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
 ( ) [inline]

6.10.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
 ( ) [inline, virtual]

### 6.10.3 Member Function Documentation



## 6.10

### activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller Class Reference 179

6.10.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.10.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.10.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2658).

6.10.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` `[virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2659).

6.10.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` `[virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2659).

## 6.10

**activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

### Class Reference

181

```
6.10.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal2  
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (  
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2660).

```
6.10.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (  
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2661).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBlobMessageMarshaller.h**

## 6.11 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 182).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.11.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 182).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.11

**activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

**Class Reference**

**183**

### 6.11.2 Constructor & Destructor Documentation

6.11.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.11.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

### 6.11.3 Member Function Documentation

6.11.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.11.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.11.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**,  
**decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2671).

6.11.3.4 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

6.11.3.5 **virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.11

**activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

### Class Reference

185

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

```
6.11.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673).

```
6.11.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2674).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQBlobMessageMarshaller.h**

## 6.12 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 186).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.12.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 186).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.12

**activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**

### Class Reference

187

#### 6.12.2 Constructor & Destructor Documentation

6.12.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.12.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

#### 6.12.3 Member Function Documentation

6.12.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

##### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.12.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

##### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.12.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**,  
**decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2667).

6.12.3.4 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

6.12.3.5 **virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.12

**activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**

### Class Reference

189

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

```
6.12.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

```
6.12.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQBlobMessageMarshaller.h**

## 6.13 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 190).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMes
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.13.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 190).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

## 6.13

**activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

### Class Reference

191

#### 6.13.2 Constructor & Destructor Documentation

6.13.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.13.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

#### 6.13.3 Member Function Documentation

6.13.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

##### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.13.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

##### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.13.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**,  
**decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

6.13.3.4 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

6.13.3.5 **virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.13

**activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

### Class Reference

193

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.13.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.13.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2665).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBlobMessageMarshaller.h**

## 6.14 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 194).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMes
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.14.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 194).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module



## 6.14

**activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

**Class Reference**

**195**

### 6.14.2 Constructor & Destructor Documentation

6.14.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.14.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

### 6.14.3 Member Function Documentation

6.14.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.14.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.14.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**,  
**decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2654).

6.14.3.4 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.14.3.5 **virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.14

**activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

### Class Reference

197

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

```
6.14.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

```
6.14.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQBlobMessageMarshaller.h**

## 6.15 activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 198).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller:

### Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.15.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 198).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.15

**activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**

### Class Reference

199

#### 6.15.2 Constructor & Destructor Documentation

6.15.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller**  
( ) [inline]

6.15.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller**  
( ) [inline, virtual]

#### 6.15.3 Member Function Documentation

6.15.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

##### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.15.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

##### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.15.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**,  
**decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** )  
[virtual]

Write a object instance to data output stream.

##### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

6.15.3.4 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

6.15.3.5 **virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.15

**activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**

### Class Reference

201

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

```
6.15.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

```
6.15.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2678).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQBlobMessageMarshaller.h**

## 6.16 activemq::commands::ActiveMQBytesMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQBytesMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQBytesMessage:

### Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual **~ActiveMQBytesMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQBytesMessage \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **clearBody** () throw ( cms::CMSEException )  
*Clears out the body of the message.*
- virtual void **onSend** ()  
*Store the Data that was written to the stream before a send.*
- virtual void **setBodyBytes** (const unsigned char \*buffer, int numBytes) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*sets the bytes given to the message body.*
- virtual unsigned char \* **getBodyBytes** () const throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.*
- virtual int **getBodyLength** () const throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Returns the number of bytes contained in the body of this message.*
- virtual void **reset** () throw ( cms::MessageFormatException, cms::CMSEException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*



- virtual bool **readBoolean** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Boolean from the Bytes message stream.*
- virtual void **writeBoolean** (bool value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a boolean to the bytes message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Bytes message stream.*
- virtual void **writeByte** (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the bytes message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the bytes message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a portion of the bytes message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a portion of a byte array to the bytes message stream.*
- virtual char **readChar** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Char from the Bytes message stream.*
- virtual void **writeChar** (char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a char to the bytes message stream as a 1-byte value.*
- virtual float **readFloat** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit float from the Bytes message stream.*
- virtual void **writeFloat** (float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a float to the bytes message stream as a 4 byte value.*
- virtual double **readDouble** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit double from the Bytes message stream.*
- virtual void **writeDouble** (double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a double to the bytes message stream as a 8 byte value.*
- virtual short **readShort** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit signed short from the Bytes message stream.*

- virtual void **writeShort** (short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed short to the bytes message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit unsigned short from the Bytes message stream.*

- virtual void **writeUnsignedShort** (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a unsigned short to the bytes message stream as a 2 byte value.*

- virtual int **readInt** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit signed integer from the Bytes message stream.*

- virtual void **writeInt** (int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed int to the bytes message stream as a 4 byte value.*

- virtual long long **readLong** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit long from the Bytes message stream.*

- virtual void **writeLong** (long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a long long to the bytes message stream as a 8 byte value.*

- virtual std::string **readString** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an ASCII String from the Bytes message stream.*

- virtual void **writeString** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an ASCII String to the Bytes message stream.*

- virtual std::string **readUTF** () const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an UTF String from the BytesMessage stream.*

- virtual void **writeUTF** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an UTF String to the BytesMessage stream.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQBYTESMESSAGE** = 24

## 6.16.1 Constructor & Destructor Documentation

### 6.16.1.1 activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ( )

6.16.1.2 `virtual activemq::commands::ActiveMQBytesMessage::~~ActiveMQBytesMessage ( )`  
[virtual]

## 6.16.2 Member Function Documentation

6.16.2.1 `virtual void activemq::commands::ActiveMQBytesMessage::clearBody ( ) throw ( cms::CMSException )` [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 398).

6.16.2.2 `virtual cms::BytesMessage* activemq::commands::ActiveMQBytesMessage::clone ( ) const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

### Returns

new copy of this message

Implements `cms::BytesMessage` (p. 1026).

6.16.2.3 `virtual ActiveMQBytesMessage* activemq::commands::ActiveMQBytesMessage::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2480).

6.16.2.4 `virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2481).

6.16.2.5 `virtual bool activemq::commands::ActiveMQBytesMessage::equals ( const  
DataSetructure * value ) const` [virtual]

Compares the **DataSetructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Parameters

<i>value</i>	The <b>Command</b> (p. 1165) to compare to this one.
--------------	--

#### Returns

true if DataSetructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 399).

6.16.2.6 `virtual unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes ( ) const throw ( cms::MessageNotReadableException, cms::CMSException )`  
[virtual]

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.

Call `getBodyLength` to determine the number of bytes to expect.

#### Returns

const pointer to a byte buffer

#### Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>MessageNotReadableException</i>	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 1027).

6.16.2.7 `virtual int activemq::commands::ActiveMQBytesMessage::getBodyLength ( ) const  
throw ( cms::MessageNotReadableException, cms::CMSException )`  
[virtual]

Returns the number of bytes contained in the body of this message.

#### Returns

number of bytes.

**Exceptions**

<i>CMSException</i>	- If an internal error occurs.
<i>MessageNotReadableException</i>	- If the message is in Write Only Mode.

Implements **cms::BytesMessage** (p. 1027).

**6.16.2.8** `virtual unsigned char activemq::commands::ActiveMQBytesMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Message** (p. 2483).

**6.16.2.9** `virtual void activemq::commands::ActiveMQBytesMessage::onSend ( ) [virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 407).

**6.16.2.10** `virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a Boolean from the Bytes message stream.

**Returns**

boolean value from stream

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1027).

```
6.16.2.11 virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte
( ) const throw ( cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException )
[virtual]
```

Reads a Byte from the Bytes message stream.

#### Returns

unsigned char value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1028).

```
6.16.2.12 virtual int activemq::commands::ActiveMQBytesMessage::readBytes ( std::vector<
unsigned char > & value ) const throw ( cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException )
[virtual]
```

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

#### Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.

<i>MessageNotReadableException</i>	- if the message is in write-only mode.
------------------------------------	---

Implements **cms::BytesMessage** (p. 1028).

```
6.16.2.13 virtual int activemq::commands::ActiveMQBytesMessage::readBytes ( unsigned
char * buffer, int length ) const throw ( cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException )
[virtual]
```

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

#### Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1029).

6.16.2.14 `virtual char activemq::commands::ActiveMQBytesMessage::readChar ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a Char from the Bytes message stream.

#### Returns

char value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1030).

6.16.2.15 `virtual double activemq::commands::ActiveMQBytesMessage::readDouble ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a 64 bit double from the Bytes message stream.

#### Returns

double value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1030).

6.16.2.16 `virtual float activemq::commands::ActiveMQBytesMessage::readFloat ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a 32 bit float from the Bytes message stream.



**Returns**

double value from stream

**Exceptions**

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1031).

6.16.2.17 `virtual int activemq::commands::ActiveMQBytesMessage::readInt ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

**Returns**

int value from stream

**Exceptions**

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1031).

6.16.2.18 `virtual long long activemq::commands::ActiveMQBytesMessage::readLong ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]`

Reads a 64 bit long from the Bytes message stream.

**Returns**

long long value from stream

**Exceptions**

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
----------------------	---

<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1032).

6.16.2.19 `virtual short activemq::commands::ActiveMQBytesMessage::readShort  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads a 16 bit signed short from the Bytes message stream.

#### Returns

short value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1032).

6.16.2.20 `virtual std::string activemq::commands::ActiveMQBytesMessage::readString  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSException )  
[virtual]`

Reads an ASCII String from the Bytes message stream.

#### Returns

String from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1033).

6.16.2.21 virtual unsigned short activemq::commands::ActiveMQBytesMessage::readUnsignedShort  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException )  
[virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

#### Returns

unsigned short value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1033).

6.16.2.22 virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF  
( ) const throw ( cms::MessageEOFException,  
cms::MessageNotReadableException, cms::CMSEException )  
[virtual]

Reads an UTF String from the BytesMessage stream.

#### Returns

String from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1034).

6.16.2.23 `virtual void activemq::commands::ActiveMQBytesMessage::reset ( ) throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

#### Exceptions

<i>CMSException</i>	- If the provider fails to perform the reset operation.
<i>MessageFormatException</i>	- If the <b>Message</b> (p. 2475) has an invalid format.

Implements **cms::BytesMessage** (p. 1034).

6.16.2.24 `virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes ( const unsigned char * buffer, int numBytes ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

sets the bytes given to the message body.

#### Parameters

<i>buffer</i>	Byte Buffer to copy
<i>numBytes</i>	Number of bytes in Buffer to copy

#### Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>MessageNotWriteableException</i>	- if in Read Only Mode.

Implements **cms::BytesMessage** (p. 1034).

6.16.2.25 `virtual std::string activemq::commands::ActiveMQBytesMessage::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2490).

```
6.16.2.26 virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean ( bool value
) throw ( cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

#### Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1035).

```
6.16.2.27 virtual void activemq::commands::ActiveMQBytesMessage::writeByte ( unsigned char
value ) throw ( cms::MessageNotWriteableException, cms::CMSException
) [virtual]
```

Writes a byte to the bytes message stream as a 1-byte value.

#### Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1035).

```
6.16.2.28 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes
( const unsigned char * value, int offset, int length ) throw (
cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Writes a portion of a byte array to the bytes message stream.

size as the number of bytes to write.

**Parameters**

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1036).

```
6.16.2.29 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes
( const std::vector< unsigned char > & value ) throw (
  cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

**Parameters**

<i>value</i>	bytes to write to the stream
--------------	------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1036).

```
6.16.2.30 virtual void activemq::commands::ActiveMQBytesMessage::writeChar ( char value )
throw ( cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Writes a char to the bytes message stream as a 1-byte value.

**Parameters**

<i>value</i>	char to write to the stream
--------------	-----------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
---------------------	--

<i>MessageNotWriteableException</i>	- if the message is in read-only mode.
-------------------------------------	--

Implements **cms::BytesMessage** (p. 1037).

```
6.16.2.31 virtual void activemq::commands::ActiveMQBytesMessage::writeDouble ( double
        value ) throw ( cms::MessageNotWriteableException, cms::CMSException
        ) [virtual]
```

Writes a double to the bytes message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1037).

```
6.16.2.32 virtual void activemq::commands::ActiveMQBytesMessage::writeFloat ( float value )
        throw ( cms::MessageNotWriteableException, cms::CMSException )
        [virtual]
```

Writes a float to the bytes message stream as a 4 byte value.

#### Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1038).

6.16.2.33 `virtual void activemq::commands::ActiveMQBytesMessage::writeInt ( int value )  
throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[virtual]`

Writes a signed int to the bytes message stream as a 4 byte value.

#### Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1038).

6.16.2.34 `virtual void activemq::commands::ActiveMQBytesMessage::writeLong ( long long  
value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[virtual]`

Writes a long long to the bytes message stream as a 8 byte value.

#### Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1038).

6.16.2.35 `virtual void activemq::commands::ActiveMQBytesMessage::writeShort ( short value )  
throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[virtual]`

Writes a signed short to the bytes message stream as a 2 byte value.

#### Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------



**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1039).

**6.16.2.36** virtual void activemq::commands::ActiveMQBytesMessage::writeString ( const std::string & *value* ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]

Writes an ASCII String to the Bytes message stream.

**Parameters**

<i>value</i>	String to write to the stream
--------------	-------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1039).

**6.16.2.37** virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort ( unsigned short *value* ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

**Parameters**

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::BytesMessage** (p. 1040).

6.16.2.38 `virtual void activemq::commands::ActiveMQBytesMessage::writeUTF ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes an UTF String to the BytesMessage stream.

#### Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of bytes stream has been reached.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::BytesMessage** (p. 1040).

### 6.16.3 Field Documentation

6.16.3.1 `const unsigned char activemq::commands::ActiveMQBytesMessage::ID_ - ACTIVEMQBYTESMESSAGE = 24 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBytesMessage.h`

## 6.17 `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 220).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMe
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`:

#### Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const

## 6.17 ac-

### activemq:wireformat:openwire::marshal:v3::ActiveMQBytesMessageMarshaller

#### Class Reference

221

*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.17.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 220).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.17.2 Constructor & Destructor Documentation

6.17.2.1 **activemq:wireformat:openwire::marshal:v3::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.17.2.2 **virtual activemq:wireformat:openwire::marshal:v3::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

### 6.17.3 Member Function Documentation

6.17.3.1 **virtual commands::DataStructure\* ac-**  
**activemq:wireformat:openwire::marshal:v3::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.17.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.17.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2658).

```
6.17.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.17 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller

### Class Reference 223

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

**6.17.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

**6.17.3.6** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2660).

6.17.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2661).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h`

## 6.18 **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 224).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMe
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**:

**Public Member Functions**

- **ActiveMQBytesMessageMarshaller** ()

## 6.18 ac-

**ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

### Class Reference

225

- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.18.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 224).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.18.2 Constructor & Destructor Documentation

6.18.2.1 **ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.18.2.2 **virtual ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

### 6.18.3 Member Function Documentation

6.18.3.1 **virtual commands::DataStructure\* ac-**  
**ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.18.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.18.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2671).

```
6.18.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.18 activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller

### Class Reference 227

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

**6.18.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

**6.18.3.6** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673).

```
6.18.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2674).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQBytesMessageMarshaller.h**

## 6.19 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 228).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMe
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**:

**Public Member Functions**

- **ActiveMQBytesMessageMarshaller** ()

## 6.19 ac-

### ativemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

#### Class Reference

229

- virtual `~ActiveMQBytesMessageMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.19.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 228).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.19.2 Constructor & Destructor Documentation

6.19.2.1 `ativemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller ( ) [inline]`

6.19.2.2 `virtual ativemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller ( ) [inline, virtual]`

### 6.19.3 Member Function Documentation

6.19.3.1 `virtual commands::DataStructure* ativemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.19.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.19.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2667).

```
6.19.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.19 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

### Class Reference 231

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

**6.19.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

**6.19.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

```
6.19.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQBytesMessageMarshaller.h**

## 6.20 **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 232).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMe
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**:

**Public Member Functions**

- **ActiveMQBytesMessageMarshaller** ()

## 6.20 ac-

**ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**

### Class Reference

233

- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.20.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 232).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.20.2 Constructor & Destructor Documentation

6.20.2.1 **ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.20.2.2 **virtual ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

## 6.20.3 Member Function Documentation

6.20.3.1 **virtual commands::DataStructure\* ac-**  
**ativemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.20.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.20.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2654).

```
6.20.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.20 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller

### Class Reference 235

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.20.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.20.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

```
6.20.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQBytesMessageMarshaller.h**

## 6.21 **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 236).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMe
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**:

**Public Member Functions**

- **ActiveMQBytesMessageMarshaller** ()

## 6.21 ac-

**ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

### Class Reference

237

- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.21.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 236).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.21.2 Constructor & Destructor Documentation

6.21.2.1 **ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.21.2.2 **virtual ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

### 6.21.3 Member Function Documentation

6.21.3.1 **virtual commands::DataStructure\* ac-**  
**ativemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.21.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.21.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

```
6.21.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.21 activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller

### Class Reference 239

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

6.21.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

6.21.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

```
6.21.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2678).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQBytesMessageMarshaller.h**

## 6.22 **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 240).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMe
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**:

**Public Member Functions**

- **ActiveMQBytesMessageMarshaller** ()

## 6.22 ac-

**ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

### Class Reference

241

- virtual **~ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.22.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 240).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.22.2 Constructor & Destructor Documentation

6.22.2.1 **ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller**  
( ) [inline]

6.22.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller**  
( ) [inline, virtual]

### 6.22.3 Member Function Documentation

6.22.3.1 **virtual commands::DataStructure\* ac-**  
**ativemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.22.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.22.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

```
6.22.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.22 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller

### Class Reference 243

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

6.22.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

6.22.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.22.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2665).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBytesMessageMarshaller.h**

**6.23 activemq::core::ActiveMQConnection Class Reference**

Concrete connection used for all connectors to the ActiveMQ broker.

```
#include <src/main/activemq/core/ActiveMQConnection.h>
```

Inheritance diagram for **activemq::core::ActiveMQConnection**:

**Public Member Functions**

- **ActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)  
*Constructor.*

- virtual `~ActiveMQConnection ()`
- virtual void **removeSession** (**ActiveMQSession** \*session) throw ( cms::CMSException )  
*Removes the session resources for the given session instance.*
- virtual void **addProducer** (**ActiveMQProducer** \*producer) throw ( cms::CMSException )  
*Adds an active Producer to the Set of known producers.*
- virtual void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw ( cms::CMSException )  
*Removes an active Producer to the Set of known producers.*
- virtual void **addDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** \*dispatcher) throw ( cms::CMSException )  
*Adds a dispatcher for a consumer.*
- virtual void **removeDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer) throw ( cms::CMSException )  
*Removes the dispatcher for a consumer.*
- virtual void **sendPullRequest** (const **commands::ConsumerInfo** \*consumer, long long timeout) throw ( exceptions::ActiveMQException )  
*If supported sends a message pull request to the service provider asking for the delivery of a new message.*
- bool **isClosed** () const  
*Checks if this connection has been closed.*
- bool **isStarted** () const  
*Check if this connection has been started.*
- bool **isTransportFailed** () const  
*Checks if the Connection's Transport has failed.*
- virtual void **destroyDestination** (const **commands::ActiveMQDestination** \*destination) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException )  
*Requests that the Broker removes the given Destination.*
- virtual void **destroyDestination** (const **cms::Destination** \*destination) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException )  
*Requests that the Broker removes the given Destination.*
- virtual const **cms::ConnectionMetaData** \* **getMetaData** () const throw ( cms::CMSException )  
*Gets the metadata for this connection.*
- virtual **cms::Session** \* **createSession** () throw ( cms::CMSException )  
*Creates a new Session to work for this Connection.*
- virtual std::string **getClientID** () const  
*Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.*

**Returns**

Client Id String for this **Connection** (p. 1232).

**Exceptions**

<b>CMSEException</b> (p. 1130)	if the provider fails to return the client id or an internal error occurs.
-----------------------------------	--

- virtual void **setClientID** (const std::string &clientID)

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1294) object and transparently assigned to the **Connection** (p. 1232) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1958).

**Parameters**

clientID	The unique client identifier to assign to the <b>Connection</b> (p. 1232).
----------	--

**Exceptions**

<b>CMSEException</b> (p. 1130)	if the provider fails to set the client id due to some internal error.
<b>InvalidClientIDException</b>	if the id given is somehow invalid or is a duplicate.
<b>IllegalStateException</b> (p. 1958)	if the client tries to set the id after a <b>Connection</b> (p. 1232) method has been called.

- virtual **cms::Session \* createSession** (**cms::Session::AcknowledgeMode** ackMode) throw ( cms::CMSEException )

Creates a new Session to work for this Connection using the specified acknowledgment mode.

- virtual void **close** () throw ( cms::CMSEException )

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

- virtual void **start** () throw ( cms::CMSEException )

Starts or (restarts) a connections delivery of incoming messages.

- virtual void **stop** () throw ( cms::CMSEException )

Stop the flow of incoming messages.

- virtual **cms::ExceptionListener \* getExceptionListener** () const

Gets the registered Exception Listener for this connection.

- virtual void **setExceptionListener** (**cms::ExceptionListener \*listener**)

Sets the registered Exception Listener for this connection.

- void **setUsername** (const std::string &username)

Sets the username that should be used when creating a new connection.

- const std::string & **getUsername** () const

Gets the username that this factory will use when creating a new connection instance.

- void **setPassword** (const std::string &password)

Sets the password that should be used when creating a new connection.

- const std::string & **getPassword** () const

*Gets the password that this factory will use when creating a new connection instance.*

- void **setDefaultClientId** (const std::string &clientId)

*Sets the Client Id.*

- void **setBrokerURL** (const std::string &brokerURL)

*Sets the Broker URL that should be used when creating a new connection instance.*

- const std::string & **getBrokerURL** () const

*Gets the Broker URL that this factory will use when creating a new connection instance.*

- void **setPrefetchPolicy** (**PrefetchPolicy** \*policy)

*Sets the **PrefetchPolicy** (p. 2924) instance that this factory should use when it creates new Connection instances.*

- **PrefetchPolicy** \* **getPrefetchPolicy** () const

*Gets the pointer to the current **PrefetchPolicy** (p. 2924) that is in use by this ConnectionFactory.*

- void **setRedeliveryPolicy** (**RedeliveryPolicy** \*policy)

*Sets the **RedeliveryPolicy** (p. 3121) instance that this factory should use when it creates new Connection instances.*

- **RedeliveryPolicy** \* **getRedeliveryPolicy** () const

*Gets the pointer to the current **RedeliveryPolicy** (p. 3121) that is in use by this ConnectionFactory.*

- bool **isDispatchAsync** () const

- void **setDispatchAsync** (bool value)

*Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.*

- bool **isAlwaysSyncSend** () const

*Gets if the Connection should always send things Synchronously.*

- void **setAlwaysSyncSend** (bool value)

*Sets if the Connection should always send things Synchronously.*

- bool **isUseAsyncSend** () const

*Gets if the useAsyncSend option is set.*

- void **setUseAsyncSend** (bool value)

*Sets the useAsyncSend option.*

- bool **isUseCompression** () const

*Gets if the Connection is configured for Message body compression.*

- void **setUseCompression** (bool value)

*Sets whether Message body compression is enabled.*

- unsigned int **getSendTimeout** () const

*Gets the assigned send timeout for this Connector.*

- void **setSendTimeout** (unsigned int timeout)

*Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.*

- unsigned int **getCloseTimeout** () const

*Gets the assigned close timeout for this Connector.*

- void **setCloseTimeout** (unsigned int timeout)

*Sets the close timeout to use when sending the disconnect request.*

- unsigned int **getProducerWindowSize** () const  
*Gets the configured producer window size for Producers that are created from this connector.*
- void **setProducerWindowSize** (unsigned int windowSize)  
*Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.*
- long long **getNextSessionId** ()  
*Get the Next available Session Id.*
- long long **getNextTempDestinationId** ()  
*Get the Next Temporary Destination Id.*
- long long **getNextLocalTransactionId** ()  
*Get the Next Temporary Destination Id.*
- void **addTransportListener** (transport::TransportListener \*transportListener)  
  
*Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.*
- void **removeTransportListener** (transport::TransportListener \*transportListener)  
  
*Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.*
- virtual void **onCommand** (const Pointer< commands::Command > &command)  
*Event handler for the receipt of a non-response command from the transport.*
- virtual void **onException** (const decaf::lang::Exception &ex)  
*Event handler for an exception from a command transport.*
- virtual void **transportInterrupted** ()  
*The transport has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
*The transport has resumed after an interruption.*
- const **commands::ConnectionInfo** & **getConnectionInfo** () const throw ( exceptions::ActiveMQException )  
*Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.*
- const **commands::ConnectionId** & **getConnectionId** () const throw ( exceptions::ActiveMQException )  
*Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.*
- **transport::Transport** & **getTransport** () const  
*Gets a reference to this object's Transport instance.*
- void **oneway** (Pointer< commands::Command > command) throw ( activemq::exceptions::ActiveMQException )  
*Sends a oneway message.*
- void **syncRequest** (Pointer< commands::Command > command, unsigned int timeout=0) throw ( activemq::exceptions::ActiveMQException )

*Sends a synchronous request and returns the response from the broker.*

- virtual void **fire** (const **exceptions::ActiveMQException** &ex)

*Notify the exception listener.*

- void **setTransportInterruptionProcessingComplete** ()

*Indicates that a Connection resource that is processing the transportInterrupted event has completed.*

### 6.23.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

#### Since

2.0

### 6.23.2 Constructor & Destructor Documentation

- 6.23.2.1 **activemq::core::ActiveMQConnection::ActiveMQConnection** ( const **Pointer**< **transport::Transport** > & *transport*, const **Pointer**< **decaf::util::Properties** > & *properties* )

Constructor.

#### Parameters

<i>transport</i>	The Transport requested for this connection to the Broker.
<i>properties</i>	The Properties that were defined for this connection

- 6.23.2.2 **virtual activemq::core::ActiveMQConnection::~~ActiveMQConnection** ( )  
[virtual]

### 6.23.3 Member Function Documentation

- 6.23.3.1 **virtual void activemq::core::ActiveMQConnection::addDispatcher** ( const **Pointer**< **commands::ConsumerId** > & *consumer*, **Dispatcher** \* *dispatcher* ) throw ( **cms::CMSException** ) [virtual]

Adds a dispatcher for a consumer.

#### Parameters

<i>consumer</i>	- The consumer for which to register a dispatcher.
<i>dispatcher</i>	- The dispatcher to handle incoming messages for the consumer.

6.23.3.2 `virtual void activemq::core::ActiveMQConnection::addProducer (   
 ActiveMQProducer * producer ) throw ( cms::CMSEException ) [virtual]`

Adds an active Producer to the Set of known producers.

#### Parameters

<i>producer</i>	- The Producer to add from the the known set.
-----------------	---

6.23.3.3 `void activemq::core::ActiveMQConnection::addTransportListener (   
 transport::TransportListener * transportListener )`

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.

Client's should ensure that the registered listener does not block or take a long amount of time to execute in order to not degrade performance of this Connection.

#### Parameters

<i>transportListener</i>	The TransportListener instance to add to this Connection's set of listeners to notify of Transport events.
--------------------------	--

6.23.3.4 `virtual void activemq::core::ActiveMQConnection::close ( ) throw (   
 cms::CMSEException ) [virtual]`

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

#### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Connection** (p. 1234).

6.23.3.5 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession (   
 cms::Session::AcknowledgeMode ackMode ) throw ( cms::CMSEException )   
 [virtual]`

Creates a new Session to work for this Connection using the specified acknowledgment mode.

#### Parameters

<i>ackMode</i>	the Acknowledgment Mode to use.
----------------	---------------------------------



**Exceptions**

<i>CMSException</i>	
---------------------	--

Implements **cms::Connection** (p. 1234).

6.23.3.6 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession ( )  
throw ( cms::CMSException ) [virtual]`

Creates a new Session to work for this Connection.

**Exceptions**

<i>CMSException</i>	
---------------------	--

Implements **cms::Connection** (p. 1235).

6.23.3.7 `virtual void activemq::core::ActiveMQConnection::destroyDestination  
( const commands::ActiveMQDestination * destination )  
throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalStateException,  
decaf::lang::exceptions::UnsupportedOperationException,  
activemq::exceptions::ActiveMQException ) [virtual]`

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

**Parameters**

<i>destination</i>	The Destination the Broker will be requested to remove.
--------------------	---

**Exceptions**

<i>NullPointerException</i>	If the passed Destination is Null
<i>IllegalStateException</i>	If the connection is closed.
<i>UnsupportedOperationException</i>	If the wire format in use does not support this operation.
<i>ActiveMQException</i>	If any other error occurs during the attempt to destroy the destination.

```
6.23.3.8 virtual void activemq::core::ActiveMQConnection::destroyDestination
( const cms::Destination * destination ) throw (
    decaf::lang::exceptions::NullPointerException,
    decaf::lang::exceptions::IllegalStateException,
    decaf::lang::exceptions::UnsupportedOperationException,
    activemq::exceptions::ActiveMQException ) [virtual]
```

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

#### Parameters

<i>destination</i>	The CMS Destination the Broker will be requested to remove.
--------------------	---

#### Exceptions

<i>NullPointerException</i>	If the passed Destination is Null
<i>IllegalStateException</i>	If the connection is closed.
<i>UnsupportedOperationException</i>	If the wire format in use does not support this operation.
<i>ActiveMQException</i>	If any other error occurs during the attempt to destroy the destination.

```
6.23.3.9 virtual void activemq::core::ActiveMQConnection::fire ( const
    exceptions::ActiveMQException & ex ) [virtual]
```

Notify the exception listener.

#### Parameters

<i>ex</i>	the exception to fire
-----------	-----------------------

```
6.23.3.10 const std::string& activemq::core::ActiveMQConnection::getBrokerURL ( ) const
```

Gets the Broker URL that this factory will use when creating a new connection instance.

#### Returns

brokerURL string

```
6.23.3.11 virtual std::string activemq::core::ActiveMQConnection::getClientID ( ) const
[virtual]
```

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

**Returns**

Client Id String for this **Connection** (p. 1232).

**Exceptions**

<b>CMSException</b> (p. 1130)	if the provider fails to return the client id or an internal error occurs.
----------------------------------	--

Implements **cms::Connection** (p. 1235).

6.23.3.12 `unsigned int activemq::core::ActiveMQConnection::getCloseTimeout ( ) const`

Gets the assigned close timeout for this Connector.

**Returns**

the close timeout configured in the connection uri

6.23.3.13 `const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId ( ) const`  
`throw ( exceptions::ActiveMQException )`

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

6.23.3.14 `const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo ( )`  
`const throw ( exceptions::ActiveMQException )`

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

6.23.3.15 `virtual cms::ExceptionListener* activemq::core::ActiveMQConnection::getExceptionListener ( )`  
`const [virtual]`

Gets the registered Exception Listener for this connection.

**Returns**

pointer to an exception listener or NULL

Implements **cms::Connection** (p. 1235).

6.23.3.16 `virtual const cms::ConnectionMetaData*`  
`activemq::core::ActiveMQConnection::getMetaData ( ) const throw (`  
`cms::CMSException ) [inline, virtual]`

Gets the metadata for this connection.

#### Returns

the connection MetaData pointer ( caller does not own it ).

#### Exceptions

<i>CMSException</i>	if the provider fails to get the connection metadata for this connection.
---------------------	---

#### See also

ConnectionMetaData

#### Since

2.0

Implements **cms::Connection** (p. 1235).

6.23.3.17 `long long activemq::core::ActiveMQConnection::getNextLocalTransactionId ( )`

Get the Next Temporary Destination Id.

#### Returns

the next id in the sequence.

6.23.3.18 `long long activemq::core::ActiveMQConnection::getNextSessionId ( )`

Get the Next available Session Id.

#### Returns

the next id in the sequence.

6.23.3.19 `long long activemq::core::ActiveMQConnection::getNextTempDestinationId ( )`

Get the Next Temporary Destination Id.

#### Returns

the next id in the sequence.

6.23.3.20 `const std::string& activemq::core::ActiveMQConnection::getPassword ( ) const`

Gets the password that this factory will use when creating a new connection instance.

#### Returns

password string, "" for default credentials

6.23.3.21 `PrefetchPolicy* activemq::core::ActiveMQConnection::getPrefetchPolicy ( ) const`

Gets the pointer to the current **PrefetchPolicy** (p. 2924) that is in use by this ConnectionFactory.

#### Returns

a pointer to this objects **PrefetchPolicy** (p. 2924).

6.23.3.22 `unsigned int activemq::core::ActiveMQConnection::getProducerWindowSize ( ) const`

Gets the configured producer window size for Producers that are created from this connector.

This only applies if there is no send timeout and the producer is able to send asynchronously.

#### Returns

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.23.3.23 `RedeliveryPolicy* activemq::core::ActiveMQConnection::getRedeliveryPolicy ( ) const`

Gets the pointer to the current **RedeliveryPolicy** (p. 3121) that is in use by this ConnectionFactory.

#### Returns

a pointer to this objects **RedeliveryPolicy** (p. 3121).

6.23.3.24 `unsigned int activemq::core::ActiveMQConnection::getSendTimeout ( ) const`

Gets the assigned send timeout for this Connector.

#### Returns

the send timeout configured in the connection uri

**6.23.3.25** `transport::Transport& activemq::core::ActiveMQConnection::getTransport ( ) const`

Gets a reference to this object's Transport instance.

**Returns**

a reference to the Transport that is in use by this Connection.

**6.23.3.26** `const std::string& activemq::core::ActiveMQConnection::getUsername ( ) const`

Gets the username that this factory will use when creating a new connection instance.

**Returns**

username string, "" for default credentials

**6.23.3.27** `bool activemq::core::ActiveMQConnection::isAlwaysSyncSend ( ) const`

Gets if the Connection should always send things Synchronously.

**Returns**

true if sends should always be Synchronous.

**6.23.3.28** `bool activemq::core::ActiveMQConnection::isClosed ( ) const` `[inline]`

Checks if this connection has been closed.

**Returns**

true if the connection is closed

**6.23.3.29** `bool activemq::core::ActiveMQConnection::isDispatchAsync ( ) const`

**Returns**

The value of the dispatch asynchronously option sent to the broker.

**6.23.3.30** `bool activemq::core::ActiveMQConnection::isStarted ( ) const` `[inline]`

Check if this connection has been started.

**Returns**

true if the start method has been called.

6.23.3.31 `bool activemq::core::ActiveMQConnection::isTransportFailed ( ) const`  
`[inline]`

Checks if the Connection's Transport has failed.

#### Returns

true if the Connection's Transport has failed.

6.23.3.32 `bool activemq::core::ActiveMQConnection::isUseAsyncSend ( ) const`

Gets if the useAsyncSend option is set.

#### Returns

true if on false if not.

6.23.3.33 `bool activemq::core::ActiveMQConnection::isUseCompression ( ) const`

Gets if the Connection is configured for Message body compression.

#### Returns

if the Message body will be Compressed or not.

6.23.3.34 `virtual void activemq::core::ActiveMQConnection::onCommand ( const Pointer<`  
`commands::Command > & command ) [virtual]`

Event handler for the receipt of a non-response command from the transport.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements **activemq::transport::TransportListener** (p. 3836).

6.23.3.35 `void activemq::core::ActiveMQConnection::oneway (`  
`Pointer< commands::Command > command ) throw (`  
`activemq::exceptions::ActiveMQException )`

Sends a oneway message.

#### Parameters

<i>command</i>	The message to send.
----------------	----------------------

**Exceptions**

<i>ConnectorException</i>	if not currently connected, or if the operation fails for any reason.
---------------------------	---

6.23.3.36 `virtual void activemq::core::ActiveMQConnection::onException ( const decaf::lang::Exception & ex ) [virtual]`

Event handler for an exception from a command transport.

**Parameters**

<i>ex</i>	The exception.
-----------	----------------

Implements **activemq::transport::TransportListener** (p. 3837).

6.23.3.37 `virtual void activemq::core::ActiveMQConnection::removeDispatcher ( const Pointer< commands::ConsumerId > & consumer ) throw ( cms::CMSEException ) [virtual]`

Removes the dispatcher for a consumer.

**Parameters**

<i>consumer</i>	- The consumer for which to remove the dispatcher.
-----------------	--

6.23.3.38 `virtual void activemq::core::ActiveMQConnection::removeProducer ( const Pointer< commands::ProducerId > & producerId ) throw ( cms::CMSEException ) [virtual]`

Removes an active Producer to the Set of known producers.

**Parameters**

<i>producerId</i>	- The ProducerId to remove from the the known set.
-------------------	--

6.23.3.39 `virtual void activemq::core::ActiveMQConnection::removeSession ( ActiveMQSession * session ) throw ( cms::CMSEException ) [virtual]`

Removes the session resources for the given session instance.

**Parameters**

<i>session</i>	The session to be unregistered from this connection.
----------------	--



6.23.3.40 void `activemq::core::ActiveMQConnection::removeTransportListener ( transport::TransportListener * transportListener )`

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.

The caller is responsible for freeing the listener in all cases.

#### Parameters

<i>transportListener</i>	The pointer to the TransportListener to remove from the set of listeners.
--------------------------	---

6.23.3.41 virtual void `activemq::core::ActiveMQConnection::sendPullRequest ( const commands::ConsumerInfo * consumer, long long timeout ) throw ( exceptions::ActiveMQException )` [virtual]

If supported sends a message pull request to the service provider asking for the delivery of a new message.

This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

#### Parameters

<i>consumer</i>	- the ConsumerInfo for the requesting Consumer.
<i>timeout</i>	- the time that the client is willing to wait.

6.23.3.42 void `activemq::core::ActiveMQConnection::setAlwaysSyncSend ( bool value )`

Sets if the Connection should always send things Synchronously.

#### Parameters

<i>value</i>	true if sends should always be Synchronous.
--------------	---

6.23.3.43 void `activemq::core::ActiveMQConnection::setBrokerURL ( const std::string & brokerURL )`

Sets the Broker URL that should be used when creating a new connection instance.

#### Parameters

<i>brokerURL</i>	string
------------------	--------

6.23.3.44 `virtual void activemq::core::ActiveMQConnection::setClientID ( const std::string & clientID ) [virtual]`

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1294) object and transparently assigned to the **Connection** (p. 1232) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1958).

#### Parameters

<i>clientID</i>	The unique client identifier to assign to the <b>Connection</b> (p. 1232).
-----------------	--

#### Exceptions

<b>CMSException</b> (p. 1130)	if the provider fails to set the client id due to some internal error.
<i>InvalidClientIDException</i>	if the id given is somehow invalid or is a duplicate.
<b>IllegalStateException</b> (p. 1958)	if the client tries to set the id after a <b>Connection</b> (p. 1232) method has been called.

Implements **cms::Connection** (p. 1236).

6.23.3.45 `void activemq::core::ActiveMQConnection::setCloseTimeout ( unsigned int timeout )`

Sets the close timeout to use when sending the disconnect request.

#### Parameters

<i>timeout</i>	- The time to wait for a close message.
----------------	---

6.23.3.46 `void activemq::core::ActiveMQConnection::setDefaultClientId ( const std::string & clientId )`

Sets the Client Id.

#### Parameters

<i>clientId</i>	- The new clientId value.
-----------------	---------------------------

6.23.3.47 void activemq::core::ActiveMQConnection::setDispatchAsync ( bool *value* )

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

#### Parameters

<i>value</i>	The value of the dispatch asynchronously option sent to the broker.
--------------	---

6.23.3.48 virtual void activemq::core::ActiveMQConnection::setExceptionListener ( cms::ExceptionListener \* *listener* ) [virtual]

Sets the registered Exception Listener for this connection.

#### Parameters

<i>listener</i>	pointer to and ExceptionListener
-----------------	----------------------------------

Implements **cms::Connection** (p. 1236).

6.23.3.49 void activemq::core::ActiveMQConnection::setPassword ( const std::string & *password* )

Sets the password that should be used when creating a new connection.

#### Parameters

<i>password</i>	string
-----------------	--------

6.23.3.50 void activemq::core::ActiveMQConnection::setPrefetchPolicy ( PrefetchPolicy \* *policy* )

Sets the **PrefetchPolicy** (p. 2924) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 2924) passed becomes the property of the factory and will be deleted when the factory is destroyed.

#### Parameters

<i>policy</i>	The new <b>PrefetchPolicy</b> (p. 2924) that the ConnectionFactory should clone for Connections.
---------------	--

6.23.3.51 `void activemq::core::ActiveMQConnection::setProducerWindowSize ( unsigned int windowSize )`

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

#### Parameters

<i>windowSize</i>	- The size in bytes of the Producers memory window.
-------------------	---

6.23.3.52 `void activemq::core::ActiveMQConnection::setRedeliveryPolicy ( RedeliveryPolicy * policy )`

Sets the **RedeliveryPolicy** (p. 3121) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 3121) passed becomes the property of the factory and will be deleted when the factory is destroyed.

#### Parameters

<i>policy</i>	The new <b>RedeliveryPolicy</b> (p. 3121) that the ConnectionFactory should clone for Connections.
---------------	--

6.23.3.53 `void activemq::core::ActiveMQConnection::setSendTimeout ( unsigned int timeout )`

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

#### Parameters

<i>timeout</i>	- The time to wait for a response.
----------------	------------------------------------

6.23.3.54 `void activemq::core::ActiveMQConnection::setTransportInterruptProcessingComplete ( )`

Indicates that a Connection resource that is processing the transportInterrupted event has completed.

6.23.3.55 `void activemq::core::ActiveMQConnection::setUseAsyncSend ( bool value )`

Sets the useAsyncSend option.

#### Parameters

<i>value</i>	- true to activate, false to disable.
--------------	---------------------------------------

6.23.3.56 void activemq::core::ActiveMQConnection::setUseCompression ( bool *value* )

Sets whether Message body compression is enabled.

#### Parameters

<i>value</i>	Boolean indicating if Message body compression is enabled.
--------------	--

6.23.3.57 void activemq::core::ActiveMQConnection::setUsername ( const std::string & *username* )

Sets the username that should be used when creating a new connection.

#### Parameters

<i>username</i>	string
-----------------	--------

6.23.3.58 virtual void activemq::core::ActiveMQConnection::start ( ) throw ( cms::CMSException ) [virtual]

Starts or (restarts) a connections delivery of incoming messages.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Startable** (p. 3527).

6.23.3.59 virtual void activemq::core::ActiveMQConnection::stop ( ) throw ( cms::CMSException ) [virtual]

Stop the flow of incoming messages.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Stoppable** (p. 3591).

6.23.3.60 void activemq::core::ActiveMQConnection::syncRequest ( Pointer< commands::Command > *command*, unsigned int *timeout* = 0 ) throw ( activemq::exceptions::ActiveMQException )

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

**Parameters**

<i>command</i>	The request command.
<i>timeout</i>	The time to wait for a response, default is zero or infinite.

**Exceptions**

<i>ConnectorException</i>	thrown if an error response was received from the broker, or if any other error occurred.
---------------------------	---

**6.23.3.61** `virtual void activemq::core::ActiveMQConnection::transportInterrupted ( )`  
`[virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3837).

**6.23.3.62** `virtual void activemq::core::ActiveMQConnection::transportResumed ( )`  
`[virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3837).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnection.h`

## 6.24 **activemq::core::ActiveMQConnectionFactory** Class Reference

```
#include <src/main/activemq/core/ActiveMQConnectionFactory.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnectionFactory`:

**Public Member Functions**

- **ActiveMQConnectionFactory** ( )
- **ActiveMQConnectionFactory** (const std::string &url, const std::string &username="", const std::string &password="")  
*Constructor.*
- virtual **~ActiveMQConnectionFactory** ( )
- virtual **cms::Connection \* createConnection** ( ) throw ( cms::CMSException )  
*Creates a connection with the default user identity.*
- virtual **cms::Connection \* createConnection** (const std::string &username, const std::string &password) throw ( cms::CMSException )

*Creates a connection with the specified user identity.*

- virtual **cms::Connection \* createConnection** (const std::string &username, const std::string &password, const std::string &clientId) throw ( cms::CMSException )

*Creates a connection with the specified user identity.*

- void **setUsername** (const std::string &username)

*Sets the username that should be used when creating a new connection.*

- const std::string & **getUsername** () const

*Gets the username that this factory will use when creating a new connection instance.*

- void **setPassword** (const std::string &password)

*Sets the password that should be used when creating a new connection.*

- const std::string & **getPassword** () const

*Gets the password that this factory will use when creating a new connection instance.*

- std::string **getClientId** () const

*Gets the Configured Client Id.*

- void **setClientId** (const std::string &clientId)

*Sets the Client Id.*

- void **setBrokerURL** (const std::string &brokerURL)

*Sets the Broker URL that should be used when creating a new connection instance.*

- const std::string & **getBrokerURL** () const

*Gets the Broker URL that this factory will use when creating a new connection instance.*

- void **setExceptionListener** (cms::ExceptionListener \*listener)

*Set an CMS ExceptionListener that will be set on eat connection once it has been created.*

- cms::ExceptionListener \* **getExceptionListener** () const

*Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.*

- void **setPrefetchPolicy** (PrefetchPolicy \*policy)

*Sets the **PrefetchPolicy** (p. 2924) instance that this factory should use when it creates new Connection instances.*

- PrefetchPolicy \* **getPrefetchPolicy** () const

*Gets the pointer to the current **PrefetchPolicy** (p. 2924) that is in use by this ConnectionFactory.*

- void **setRedeliveryPolicy** (RedeliveryPolicy \*policy)

*Sets the **RedeliveryPolicy** (p. 3121) instance that this factory should use when it creates new Connection instances.*

- RedeliveryPolicy \* **getRedeliveryPolicy** () const

*Gets the pointer to the current **RedeliveryPolicy** (p. 3121) that is in use by this ConnectionFactory.*

- bool **isDispatchAsync** () const

- void **setDispatchAsync** (bool value)

*Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.*

- bool **isAlwaysSyncSend** () const

*Gets if the Connection should always send things Synchronously.*

- void **setAlwaysSyncSend** (bool value)  
*Sets if the Connection should always send things Synchronously.*
- bool **isUseAsyncSend** () const  
*Gets if the useAsyncSend option is set.*
- void **setUseAsyncSend** (bool value)  
*Sets the useAsyncSend option.*
- bool **isUseCompression** () const  
*Gets if the Connection is configured for Message body compression.*
- void **setUseCompression** (bool value)  
*Sets whether Message body compression is enabled.*
- unsigned int **getSendTimeout** () const  
*Gets the assigned send timeout for this Connector.*
- void **setSendTimeout** (unsigned int timeout)  
*Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.*
- unsigned int **getCloseTimeout** () const  
*Gets the assigned close timeout for this Connector.*
- void **setCloseTimeout** (unsigned int timeout)  
*Sets the close timeout to use when sending the disconnect request.*
- unsigned int **getProducerWindowSize** () const  
*Gets the configured producer window size for Producers that are created from this connector.*
- void **setProducerWindowSize** (unsigned int windowSize)  
*Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.*

## Static Public Member Functions

- static **cms::Connection \* createConnection** (const std::string &url, const std::string &username, const std::string &password, const std::string &clientId="") throw (cms::CMSEException )  
*Creates a connection with the specified user identity.*

## Static Public Attributes

- static const std::string **DEFAULT\_URI**

## 6.24.1 Constructor & Destructor Documentation

### 6.24.1.1 activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory ( )



```
6.24.1.2  activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory ( const
          std::string & url, const std::string & username = " ", const std::string & password =
          " " )
```

Constructor.

#### Parameters

<i>url</i>	the URL of the Broker we are connecting to.
<i>username</i>	to authenticate with, defaults to ""
<i>password</i>	to authenticate with, defaults to ""

```
6.24.1.3  virtual activemq::core::ActiveMQConnectionFactory::~~ActiveMQConnectionFactory (
          ) [virtual]
```

## 6.24.2 Member Function Documentation

```
6.24.2.1  virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection
          ( ) throw ( cms::CMSEException ) [virtual]
```

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

#### Returns

a Connection Pointer

#### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::ConnectionFactory** (p. 1296).

```
6.24.2.2  virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection
          ( const std::string & username, const std::string & password ) throw (
          cms::CMSEException ) [virtual]
```

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

#### Parameters

<i>username</i>	to authenticate with
<i>password</i>	to authenticate with

**Returns**

a Connection Pointer

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::ConnectionFactory** (p. 1295).

**6.24.2.3** `static cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ( const std::string & url, const std::string & username, const std::string & password, const std::string & clientId = " " ) throw ( cms::CMSEException ) [static]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

**Parameters**

<i>url</i>	the URL of the Broker we are connecting to.
<i>username</i>	to authenticate with
<i>password</i>	to authenticate with
<i>clientId</i>	to assign to connection, defaults to ""

**Exceptions**

<i>CMSEException.</i>	
-----------------------	--

**6.24.2.4** `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection ( const std::string & username, const std::string & password, const std::string & clientId ) throw ( cms::CMSEException ) [virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

**Parameters**

<i>username</i>	to authenticate with
<i>password</i>	to authenticate with
<i>clientId</i>	to assign to connection if "" then a random client Id is created for this connection.

**Returns**

a Connection Pointer

### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::ConnectionFactory** (p. 1296).

6.24.2.5 `const std::string& activemq::core::ActiveMQConnectionFactory::getBrokerURL ( ) const`

Gets the Broker URL that this factory will use when creating a new connection instance.

### Returns

brokerURL string

6.24.2.6 `std::string activemq::core::ActiveMQConnectionFactory::getClientId ( ) const`

Gets the Configured Client Id.

### Returns

the clientId.

6.24.2.7 `unsigned int activemq::core::ActiveMQConnectionFactory::getCloseTimeout ( ) const`

Gets the assigned close timeout for this Connector.

### Returns

the close timeout configured in the connection uri

6.24.2.8 `cms::ExceptionListener* activemq::core::ActiveMQConnectionFactory::getExceptionListener ( ) const`

Returns the currently set ExceptionListener that will be set on any new Connection instance that is created by this factory.

### Returns

a pointer to a CMS ExceptionListener instance or NULL if not set.

6.24.2.9 `const std::string& activemq::core::ActiveMQConnectionFactory::getPassword ( )`  
`const`

Gets the password that this factory will use when creating a new connection instance.

#### Returns

password string, "" for default credentials

6.24.2.10 `PrefetchPolicy* activemq::core::ActiveMQConnectionFactory::getPrefetchPolicy ( )`  
`const`

Gets the pointer to the current **PrefetchPolicy** (p. 2924) that is in use by this ConnectionFactory.

#### Returns

a pointer to this objects **PrefetchPolicy** (p. 2924).

6.24.2.11 `unsigned int activemq::core::ActiveMQConnectionFactory::getProducerWindowSize ( )`  
`const`

Gets the configured producer window size for Producers that are created from this connector.

This only applies if there is no send timeout and the producer is able to send asynchronously.

#### Returns

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.24.2.12 `RedeliveryPolicy* activemq::core::ActiveMQConnectionFactory::getRedeliveryPolicy ( )`  
`const`

Gets the pointer to the current **RedeliveryPolicy** (p. 3121) that is in use by this ConnectionFactory.

#### Returns

a pointer to this objects **RedeliveryPolicy** (p. 3121).

6.24.2.13 `unsigned int activemq::core::ActiveMQConnectionFactory::getSendTimeout ( ) const`

Gets the assigned send timeout for this Connector.

#### Returns

the send timeout configured in the connection uri

6.24.2.14 `const std::string& activemq::core::ActiveMQConnectionFactory::getUsername ( ) const`

Gets the username that this factory will use when creating a new connection instance.

#### Returns

username string, "" for default credentials

6.24.2.15 `bool activemq::core::ActiveMQConnectionFactory::isAlwaysSyncSend ( ) const`

Gets if the Connection should always send things Synchronously.

#### Returns

true if sends should always be Synchronous.

6.24.2.16 `bool activemq::core::ActiveMQConnectionFactory::isDispatchAsync ( ) const`

#### Returns

The value of the dispatch asynchronously option sent to the broker.

6.24.2.17 `bool activemq::core::ActiveMQConnectionFactory::isUseAsyncSend ( ) const`

Gets if the useAsyncSend option is set.

#### Returns

true if on false if not.

6.24.2.18 `bool activemq::core::ActiveMQConnectionFactory::isUseCompression ( ) const`

Gets if the Connection is configured for Message body compression.

#### Returns

if the Message body will be Compressed or not.

6.24.2.19 `void activemq::core::ActiveMQConnectionFactory::setAlwaysSyncSend ( bool value )`

Sets if the Connection should always send things Synchronously.

#### Parameters

<i>value</i>	true if sends should always be Synchronous.
--------------	---

6.24.2.20 `void activemq::core::ActiveMQConnectionFactory::setBrokerURL ( const std::string & brokerURL )`

Sets the Broker URL that should be used when creating a new connection instance.

#### Parameters

<i>brokerURL</i>	string
------------------	--------

6.24.2.21 `void activemq::core::ActiveMQConnectionFactory::setClientId ( const std::string & clientId )`

Sets the Client Id.

#### Parameters

<i>clientId</i>	- The new clientId value.
-----------------	---------------------------

6.24.2.22 `void activemq::core::ActiveMQConnectionFactory::setCloseTimeout ( unsigned int timeout )`

Sets the close timeout to use when sending the disconnect request.

#### Parameters

<i>timeout</i>	- The time to wait for a close message.
----------------	---

6.24.2.23 `void activemq::core::ActiveMQConnectionFactory::setDispatchAsync ( bool value )`

Should messages be dispatched synchronously or asynchronously from the producer thread for non-durable topics in the broker? For fast consumers set this to false.

For slow consumers set it to true so that dispatching will not block fast consumers. .

#### Parameters

<i>value</i>	The value of the dispatch asynchronously option sent to the broker.
--------------	---

6.24.2.24 void activemq::core::ActiveMQConnectionFactory::setExceptionListener ( cms::ExceptionListener \* *listener* )

Set an CMS ExceptionListener that will be set on eat connection once it has been created.

The factory does not take ownership of this pointer, the client must ensure that its lifetime is scoped to the connection that it is applied to.

#### Parameters

<i>listener</i>	The listener to set on the connection or NULL for no listener.
-----------------	--

6.24.2.25 void activemq::core::ActiveMQConnectionFactory::setPassword ( const std::string & *password* )

Sets the password that should be used when creating a new connection.

#### Parameters

<i>password</i>	string
-----------------	--------

6.24.2.26 void activemq::core::ActiveMQConnectionFactory::setPrefetchPolicy ( PrefetchPolicy \* *policy* )

Sets the **PrefetchPolicy** (p. 2924) instance that this factory should use when it creates new Connection instances.

The **PrefetchPolicy** (p. 2924) passed becomes the property of the factory and will be deleted when the factory is destroyed.

#### Parameters

<i>policy</i>	The new <b>PrefetchPolicy</b> (p. 2924) that the ConnectionFactory should clone for Connections.
---------------	--

6.24.2.27 void activemq::core::ActiveMQConnectionFactory::setProducerWindowSize ( unsigned int *windowSize* )

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

#### Parameters

<i>windowSize</i>	- The size in bytes of the Producers memory window.
-------------------	---

6.24.2.28 `void activemq::core::ActiveMQConnectionFactory::setRedeliveryPolicy ( RedeliveryPolicy * policy )`

Sets the **RedeliveryPolicy** (p. 3121) instance that this factory should use when it creates new Connection instances.

The **RedeliveryPolicy** (p. 3121) passed becomes the property of the factory and will be deleted when the factory is destroyed.

#### Parameters

<i>policy</i>	The new <b>RedeliveryPolicy</b> (p.3121) that the ConnectionFactory should clone for Connections.
---------------	---

6.24.2.29 `void activemq::core::ActiveMQConnectionFactory::setSendTimeout ( unsigned int timeout )`

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

#### Parameters

<i>timeout</i>	- The time to wait for a response.
----------------	------------------------------------

6.24.2.30 `void activemq::core::ActiveMQConnectionFactory::setUseAsyncSend ( bool value )`

Sets the useAsyncSend option.

#### Parameters

<i>value</i>	- true to activate, false to disable.
--------------	---------------------------------------

6.24.2.31 `void activemq::core::ActiveMQConnectionFactory::setUseCompression ( bool value )`

Sets whether Message body compression is enabled.

#### Parameters

<i>value</i>	Boolean indicating if Message body compression is enabled.
--------------	--

6.24.2.32 `void activemq::core::ActiveMQConnectionFactory::setUsername ( const std::string & username )`

Sets the username that should be used when creating a new connection.



**Parameters**

<i>username</i>	string
-----------------	--------

**6.24.3 Field Documentation****6.24.3.1 const std::string activemq::core::ActiveMQConnectionFactory::DEFAULT\_URI [static]**

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionFactory.h**

**6.25 activemq::core::ActiveMQConnectionMetaData Class Reference**

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 244) class.

```
#include <src/main/activemq/core/ActiveMQConnectionMetaData.h>
```

Inheritance diagram for activemq::core::ActiveMQConnectionMetaData:

**Public Member Functions**

- **ActiveMQConnectionMetaData** ()
- virtual **~ActiveMQConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const throw ( cms::CMSEException )  
*Gets the CMS API version.*
- virtual int **getCMSMajorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS major version number.*
- virtual int **getCMSMinorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS minor version number.*
- virtual std::string **getCMSProviderName** () const throw ( cms::CMSEException )  
*Gets the CMS provider name.*
- virtual std::string **getProviderVersion** () const throw ( cms::CMSEException )  
*Gets the CMS provider version.*
- virtual int **getProviderMajorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS provider major version number.*
- virtual int **getProviderMinorVersion** () const throw ( cms::CMSEException )  
*Gets the CMS provider minor version number.*
- virtual std::vector< std::string > **getCMSXPropertyNames** () const throw ( cms::CMSEException )  
*Gets an Vector of the CMSX property names.*

### 6.25.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 244) class.

#### Since

3.0

### 6.25.2 Constructor & Destructor Documentation

6.25.2.1 `activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData ( )`

6.25.2.2 `virtual activemq::core::ActiveMQConnectionMetaData::~~ActiveMQConnectionMetaData ( ) [virtual]`

### 6.25.3 Member Function Documentation

6.25.3.1 `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion ( ) const throw ( cms::CMSEException ) [virtual]`

Gets the CMS major version number.

#### Returns

the CMS API major version number

#### Exceptions

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1356).

6.25.3.2 `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion ( ) const throw ( cms::CMSEException ) [virtual]`

Gets the CMS minor version number.

#### Returns

the CMS API minor version number

#### Exceptions

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1356).

6.25.3.3 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName ( ) const throw ( cms::CMSEException ) [virtual]`

Gets the CMS provider name.

#### Returns

the CMS provider name

#### Exceptions

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1356).

6.25.3.4 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion ( ) const throw ( cms::CMSEException ) [virtual]`

Gets the CMS API version.

#### Returns

the CMS API Version in String form.

#### Exceptions

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1357).

6.25.3.5 `virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames ( ) const throw ( cms::CMSEException ) [virtual]`

Gets an Vector of the CMSX property names.

#### Returns

an Vector of CMSX property names

#### Exceptions

<i>CMSEException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
----------------------	--

Implements **cms::ConnectionMetaData** (p. 1357).

6.25.3.6 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion ( )  
const throw ( cms::CMSException ) [virtual]`

Gets the CMS provider major version number.

#### Returns

the CMS provider major version number

#### Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 1357).

6.25.3.7 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion ( )  
const throw ( cms::CMSException ) [virtual]`

Gets the CMS provider minor version number.

#### Returns

the CMS provider minor version number

#### Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 1358).

6.25.3.8 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion ( )  
const throw ( cms::CMSException ) [virtual]`

Gets the CMS provider version.

#### Returns

the CMS provider version

#### Exceptions

<i>CMSException</i>	If the CMS Provider fails to retrieve the metadata due to some internal error.
---------------------	--

Implements **cms::ConnectionMetaData** (p. 1358).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionMetaData.h`

## 6.26 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

### Data Structures

- class **StaticInitializer**

### Public Types

- enum **TransactionState** {  
**TRANSACTION\_STATE\_BEGIN** = 0, **TRANSACTION\_STATE\_PREPARE** = 1,  
**TRANSACTION\_STATE\_COMMITONEPHASE** = 2, **TRANSACTION\_STATE\_COMMITTWO**  
**PHASE** = 3,  
**TRANSACTION\_STATE\_ROLLBACK** = 4, **TRANSACTION\_STATE\_RECOVER**  
= 5, **TRANSACTION\_STATE\_FORGET** = 6, **TRANSACTION\_STATE\_END** = 7 }
- enum **DestinationActions** { **DESTINATION\_ADD\_OPERATION** = 0, **DESTINATION\_REMOVE\_OPERATION** = 1 }
- enum **AckType** {  
**ACK\_TYPE\_DELIVERED** = 0, **ACK\_TYPE\_POISON** = 1, **ACK\_TYPE\_CONSUMED**  
= 2, **ACK\_TYPE\_REDELIVERED** = 3,  
**ACK\_TYPE\_INDIVIDUAL** = 4 }
- enum **DestinationOption** {  
**CONSUMER\_PREFETCHSIZE**, **CONSUMER\_MAXPENDINGMSGLIMIT**, **CONSUMER\_NOLOCAL**,  
**CONSUMER\_DISPATCHASYNC**,  
**CONSUMER\_RETROACTIVE**, **CONSUMER\_SELECTOR**, **CONSUMER\_EXCLUSIVE**,  
**CONSUMER\_PRIORITY**,  
**NUM\_OPTIONS** }  
*These values represent the options that can be appended to an Destination name, i.e.*
- enum **URIParam** {  
**CONNECTION\_SENDTIMEOUT**, **CONNECTION\_PRODUCERWINDOWSIZE**, **CONNECTION\_CLOSETIMEOUT**,  
**CONNECTION\_ALWAYSSEND**,  
**CONNECTION\_USEASYNCSEND**, **CONNECTION\_USECOMPRESSION**, **CONNECTION\_DISPATCHASYNC**,  
**PARAM\_USERNAME**,  
**PARAM\_PASSWORD**, **PARAM\_CLIENTID**, **NUM\_PARAMS** }  
*These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.*

## Static Public Member Functions

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption toDestinationOption** (const std::string &option)
- static const std::string & **toString** (const **URIParam** option)
- static **URIParam toURIOption** (const std::string &option)

### 6.26.1 Detailed Description

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

### 6.26.2 Member Enumeration Documentation

#### 6.26.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

***ACK\_TYPE\_DELIVERED***  
***ACK\_TYPE\_POISON***  
***ACK\_TYPE\_CONSUMED***  
***ACK\_TYPE\_REDELIVERED***  
***ACK\_TYPE\_INDIVIDUAL***

#### 6.26.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

***DESTINATION\_ADD\_OPERATION***  
***DESTINATION\_REMOVE\_OPERATION***

#### 6.26.2.3 enum activemq::core::ActiveMQConstants::DestinationOption

These values represent the options that can be appended to an Destination name, i.e.  
 /topic/foo?consumer.exclusive=true

Enumerator:

***CONSUMER\_PREFETCHSIZE***  
***CUNSUMER\_MAXPENDINGMSGLIMIT***  
***CONSUMER\_NOLOCAL***  
***CONSUMER\_DISPATCHASYNC***

*CONSUMER\_RETROACTIVE*  
*CONSUMER\_SELECTOR*  
*CONSUMER\_EXCLUSIVE*  
*CONSUMER\_PRIORITY*  
*NUM\_OPTIONS*

#### 6.26.2.4 enum activemq::core::ActiveMQConstants::TransactionState

Enumerator:

*TRANSACTION\_STATE\_BEGIN*  
*TRANSACTION\_STATE\_PREPARE*  
*TRANSACTION\_STATE\_COMMITONEPHASE*  
*TRANSACTION\_STATE\_COMMITTWOPHASE*  
*TRANSACTION\_STATE\_ROLLBACK*  
*TRANSACTION\_STATE\_RECOVER*  
*TRANSACTION\_STATE\_FORGET*  
*TRANSACTION\_STATE\_END*

#### 6.26.2.5 enum activemq::core::ActiveMQConstants::URIParam

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Enumerator:

*CONNECTION\_SENDTIMEOUT*  
*CONNECTION\_PRODUCERWINDOWSIZE*  
*CONNECTION\_CLOSETIMEOUT*  
*CONNECTION\_ALWAYS\_SYNC\_SEND*  
*CONNECTION\_USE\_ASYNC\_SEND*  
*CONNECTION\_USE\_COMPRESSION*  
*CONNECTION\_DISPATCH\_ASYNC*  
*PARAM\_USERNAME*  
*PARAM\_PASSWORD*  
*PARAM\_CLIENTID*  
*NUM\_PARAMS*

### 6.26.3 Member Function Documentation

6.26.3.1 static **DestinationOption** activemq::core::ActiveMQConstants::toDestinationOption ( const std::string & *option* ) [inline, static]

6.26.3.2 static const std::string& activemq::core::ActiveMQConstants::toString ( const **DestinationOption** *option* ) [inline, static]

6.26.3.3 static const std::string& activemq::core::ActiveMQConstants::toString ( const **URIParam** *option* ) [inline, static]

6.26.3.4 static **URIParam** activemq::core::ActiveMQConstants::toURIOption ( const std::string & *option* ) [inline, static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConstants.h**

## 6.27 activemq::core::ActiveMQConsumer Class Reference

```
#include <src/main/activemq/core/ActiveMQConsumer.h>
```

Inheritance diagram for activemq::core::ActiveMQConsumer:

### Public Member Functions

- **ActiveMQConsumer** (**ActiveMQSession** \*session, const **Pointer**< **commands::ConsumerId** > &id, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &name, const std::string &selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, **cms::MessageListener** \*listener)

*Constructor.*

- virtual ~**ActiveMQConsumer** ()
- virtual void **start** ()

*Starts the Consumer if not already started and not closed.*

- virtual void **stop** ()

*Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.*

- virtual void **close** () throw ( cms::CMSException )

*Closes the Consumer.*

- virtual **cms::Message** \* **receive** () throw ( cms::CMSException )

*Synchronously Receive a Message.*

- virtual **cms::Message** \* **receive** (int millisecs) throw ( cms::CMSException )



*Synchronously Receive a Message, time out after defined interval.*

- virtual **cms::Message \* receiveNoWait** () throw ( cms::CMSEException )  
*Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (cms::MessageListener \*listener) throw ( cms::CMSEException )  
*Sets the MessageListener that this class will send notifs on.*
- virtual **cms::MessageListener \* getMessageListener** () const throw ( cms::CMSEException )  
*Gets the MessageListener that this class will send events to.*
- virtual std::string **getMessageSelector** () const throw ( cms::CMSEException )  
*Gets this message consumer's message selector expression.*
- virtual void **acknowledge** (const Pointer< commands::MessageDispatch > &dispatch) throw ( cms::CMSEException )  
*Method called to acknowledge the message passed, called from a message when the mode is client ack.*
- virtual void **dispatch** (const Pointer< MessageDispatch > &message)  
*Called asynchronously by the session to dispatch a message.*
- void **acknowledge** () throw ( cms::CMSEException )  
*Method called to acknowledge all messages that have been received so far.*
- void **commit** () throw ( exceptions::ActiveMQException )  
*Called to Commit the current set of messages in this Transaction.*
- void **rollback** () throw ( exceptions::ActiveMQException )  
*Called to Roll back the current set of messages in this Transaction.*
- void **doClose** () throw ( exceptions::ActiveMQException )  
*Performs the actual close operation on this consumer.*
- const Pointer< commands::ConsumerInfo > & **getConsumerInfo** () const  
*Get the Consumer information for this consumer.*
- const Pointer< commands::ConsumerId > & **getConsumerId** () const  
*Get the Consumer Id for this consumer.*
- bool **isClosed** () const
- bool **isSynchronizationRegistered** () const  
*Has this Consumer Transaction **Synchronization** (p. 3659) been added to the transaction.*
- void **setSynchronizationRegistered** (bool value)  
*Sets the **Synchronization** (p. 3659) Registered state of this consumer.*
- bool **iterate** ()  
*Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.*
- void **deliverAcks** () throw ( exceptions::ActiveMQException )  
*Forces this consumer to send all pending acks to the broker.*
- void **clearMessagesInProgress** ()  
*Called on a Failover to clear any pending messages.*
- void **inProgressClearRequired** ()

*Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.*

- long long **getLastDeliveredSequenceId** () const

*Gets the currently set Last Delivered Sequence Id.*

- void **setLastDeliveredSequenceId** (long long value)

*Sets the value of the Last Delivered Sequence Id.*

- int **getMessageAvailableCount** () const

- void **setRedeliveryPolicy** (**RedeliveryPolicy** \*policy)

*Sets the **RedeliveryPolicy** (p.3121) this Consumer should use when a rollback is performed on a transacted Consumer.*

- **RedeliveryPolicy** \* **getRedeliveryPolicy** () const

*Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.*

## Protected Member Functions

- **Pointer< MessageDispatch > dequeue** (long long timeout) throw ( cms::CMSEException )

*Used by synchronous receive methods to wait for messages to come in.*

- void **beforeMessagesConsumed** (const **Pointer< commands::MessageDispatch >** &dispatch)

*Pre-consume processing.*

- void **afterMessagesConsumed** (const **Pointer< commands::MessageDispatch >** &dispatch, bool messageExpired)

*Post-consume processing.*

## 6.27.1 Constructor & Destructor Documentation

- 6.27.1.1 **activemq::core::ActiveMQConsumer::ActiveMQConsumer** ( **ActiveMQSession** \* session, const **Pointer< commands::ConsumerId >** & id, const **Pointer< commands::ActiveMQDestination >** & destination, const std::string & name, const std::string & selector, int prefetch, int maxPendingMessageCount, bool noLocal, bool browser, bool dispatchAsync, cms::MessageListener \* listener )

Constructor.

- 6.27.1.2 **virtual activemq::core::ActiveMQConsumer::~~ActiveMQConsumer** ( )  
[virtual]

## 6.27.2 Member Function Documentation

6.27.2.1 `virtual void activemq::core::ActiveMQConsumer::acknowledge ( const Pointer< commands::MessageDispatch > & dispatch ) throw ( cms::CMSEException )`  
[virtual]

Method called to acknowledge the message passed, called from a message when the mode is client ack.

#### Parameters

<i>message</i>	the Message to Acknowledge
----------------	----------------------------

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.27.2.2 `void activemq::core::ActiveMQConsumer::acknowledge ( ) throw ( cms::CMSEException )`

Method called to acknowledge all messages that have been received so far.

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.27.2.3 `void activemq::core::ActiveMQConsumer::afterMessagelsConsumed ( const Pointer< commands::MessageDispatch > & dispatch, bool messageExpired )`  
[protected]

Post-consume processing.

#### Parameters

<i>dispatch</i>	- the consumed message
<i>messageExpired</i>	- flag indicating if the message has expired.

6.27.2.4 `void activemq::core::ActiveMQConsumer::beforeMessagelsConsumed ( const Pointer< commands::MessageDispatch > & dispatch )` [protected]

Pre-consume processing.

#### Parameters

<i>dispatch</i>	- the message being consumed.
-----------------	-------------------------------

6.27.2.5 `void activemq::core::ActiveMQConsumer::clearMessagesInProgress ( )`

Called on a Failover to clear any pending messages.

6.27.2.6 `virtual void activemq::core::ActiveMQConsumer::close ( ) throw ( cms::CMSException )` [virtual]

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

#### Exceptions

<i>CMSException</i>
---------------------

Implements **cms::Closeable** (p. 1120).

6.27.2.7 `void activemq::core::ActiveMQConsumer::commit ( ) throw ( exceptions::ActiveMQException )`

Called to Commit the current set of messages in this Transaction.

#### Exceptions

<i>ActiveMQException</i>
--------------------------

6.27.2.8 `void activemq::core::ActiveMQConsumer::deliverAcks ( ) throw ( exceptions::ActiveMQException )`

Forces this consumer to send all pending acks to the broker.

6.27.2.9 `Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue ( long long timeout ) throw ( cms::CMSException )` [protected]

Used by synchronous receive methods to wait for messages to come in.

#### Parameters

<i>timeout</i>	- The maximum number of milliseconds to wait before returning. If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.
----------------	---

#### Returns

the message, if received within the allotted time. Otherwise NULL.

**Exceptions**

<i>InvalidStateException</i>	if this consumer is closed upon entering this method.
------------------------------	---

6.27.2.10 `virtual void activemq::core::ActiveMQConsumer::dispatch ( const Pointer< MessageDispatch > & message ) [virtual]`

Called asynchronously by the session to dispatch a message.

**Parameters**

<i>message</i>	dispatch object pointer
----------------	-------------------------

Implements **activemq::core::Dispatcher** (p. 1750).

6.27.2.11 `void activemq::core::ActiveMQConsumer::doClose ( ) throw ( exceptions::ActiveMQException )`

Performs the actual close operation on this consumer.

**Exceptions**

<i>ActiveMQException</i>	
--------------------------	--

6.27.2.12 `const Pointer<commands::ConsumerId>& activemq::core::ActiveMQConsumer::getConsumerId ( ) const [inline]`

Get the Consumer Id for this consumer.

**Returns**

Reference to a Consumer Id Object

6.27.2.13 `const Pointer<commands::ConsumerInfo>& activemq::core::ActiveMQConsumer::getConsumerInfo ( ) const [inline]`

Get the Consumer information for this consumer.

**Returns**

Reference to a Consumer Info Object

6.27.2.14 `long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId ( ) const [inline]`

Gets the currently set Last Delivered Sequence Id.

**Returns**

long long containing the sequence id of the last delivered Message.

6.27.2.15 `int activemq::core::ActiveMQConsumer::getMessageAvailableCount ( ) const`

**Returns**

the number of Message's this consumer is waiting to Dispatch.

6.27.2.16 `virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the MessageListener that this class will send events to.

**Returns**

the currently registered MessageListener interface pointer.

Implements **cms::MessageConsumer** (p. 2552).

6.27.2.17 `virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector ( ) const throw ( cms::CMSEException ) [virtual]`

Gets this message consumer's message selector expression.

**Returns**

This Consumer's selector expression or "".

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1130)
---

Implements **cms::MessageConsumer** (p. 2552).

6.27.2.18 `RedeliveryPolicy* activemq::core::ActiveMQConsumer::getRedeliveryPolicy ( ) const [inline]`

Gets a pointer to this Consumer's Redelivery Policy object, the Consumer retains ownership of this pointer so the caller should not delete it.

**Returns**

a Pointer to a **RedeliveryPolicy** (p. 3121) that is in use by this Consumer.

6.27.2.19 void activemq::core::ActiveMQConsumer::inProgressClearRequired ( )

Signals that a Failure occurred and that anything in-progress in the consumer should be cleared.

6.27.2.20 bool activemq::core::ActiveMQConsumer::isClosed ( ) const [inline]

#### Returns

if this Consumer has been closed.

6.27.2.21 bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered ( ) const [inline]

Has this Consumer Transaction **Synchronization** (p. 3659) been added to the transaction.

#### Returns

true if the synchronization has been added.

6.27.2.22 bool activemq::core::ActiveMQConsumer::iterate ( )

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

6.27.2.23 virtual cms::Message\* activemq::core::ActiveMQConsumer::receive ( ) throw ( cms::CMSEException ) [virtual]

Synchronously Receive a Message.

#### Returns

new message

#### Exceptions

CMSEException	
---------------	--

Implements cms::MessageConsumer (p. 2553).

6.27.2.24 virtual cms::Message\* activemq::core::ActiveMQConsumer::receive ( int millisecs ) throw ( cms::CMSEException ) [virtual]

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

#### Parameters

<i>milliseconds</i>	the time in milliseconds to wait before returning
---------------------	---

#### Returns

new message or null on timeout

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::MessageConsumer** (p. 2552).

```
6.27.2.25 virtual cms::Message* activemq::core::ActiveMQConsumer::receiveNoWait ( )  
          throw ( cms::CMSException ) [virtual]
```

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

#### Returns

new message

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::MessageConsumer** (p. 2553).

```
6.27.2.26 void activemq::core::ActiveMQConsumer::rollback ( ) throw (  
          exceptions::ActiveMQException )
```

Called to Roll back the current set of messages in this Transaction.

#### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

```
6.27.2.27 void activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId ( long long  
          value ) [inline]
```

Sets the value of the Last Delivered Sequence Id.

#### Parameters

<i>value</i>	The new value to assign to the Last Delivered Sequence Id property.
--------------	---



6.27.2.28 `virtual void activemq::core::ActiveMQConsumer::setMessageListener ( cms::MessageListener * listener ) throw ( cms::CMSException ) [virtual]`

Sets the MessageListener that this class will send notifs on.

#### Parameters

<i>listener</i>	MessageListener interface pointer
-----------------	-----------------------------------

Implements **cms::MessageConsumer** (p. 2553).

6.27.2.29 `void activemq::core::ActiveMQConsumer::setRedeliveryPolicy ( RedeliveryPolicy * policy ) [inline]`

Sets the **RedeliveryPolicy** (p. 3121) this Consumer should use when a rollback is performed on a transacted Consumer.

The Consumer takes ownership of the passed pointer. The Consumer's redelivery policy can never be null, a call to this method with a NULL pointer is ignored.

#### Parameters

<i>policy</i>	Pointer to a Redelivery Policy object that his Consumer will use.
---------------	---

6.27.2.30 `void activemq::core::ActiveMQConsumer::setSynchronizationRegistered ( bool value ) [inline]`

Sets the **Synchronization** (p. 3659) Registered state of this consumer.

#### Parameters

<i>value</i>	- true if registered false otherwise.
--------------	---------------------------------------

6.27.2.31 `virtual void activemq::core::ActiveMQConsumer::start ( ) [virtual]`

Starts the Consumer if not already started and not closed.

A consumer will no deliver messages until started.

6.27.2.32 `virtual void activemq::core::ActiveMQConsumer::stop ( ) [virtual]`

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.

A Closed Consumer is also a stopped consumer.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConsumer.h**

## 6.28 activemq::library::ActiveMQCPP Class Reference

```
#include <src/main/activemq/library/ActiveMQCPP.h>
```

### Public Member Functions

- virtual **~ActiveMQCPP** ()

### Static Public Member Functions

- static void **initializeLibrary** ()  
*Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.*
- static void **initializeLibrary** (int argc, char \*\*argv)  
*Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.*
- static void **shutdownLibrary** ()  
*Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.*

### Protected Member Functions

- **ActiveMQCPP** ()
- **ActiveMQCPP** (const **ActiveMQCPP** &)
- **ActiveMQCPP** & **operator=** (const **ActiveMQCPP** &)

#### 6.28.1 Constructor & Destructor Documentation

6.28.1.1 **activemq::library::ActiveMQCPP::ActiveMQCPP** ( ) [inline, protected]

6.28.1.2 **activemq::library::ActiveMQCPP::ActiveMQCPP** ( const **ActiveMQCPP** & )  
[protected]

6.28.1.3 **virtual activemq::library::ActiveMQCPP::~~ActiveMQCPP** ( ) [inline, virtual]

#### 6.28.2 Member Function Documentation

6.28.2.1 **static void activemq::library::ActiveMQCPP::initializeLibrary** ( ) [static]

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.

**Exceptions**

<i>runtime_error</i>	if an error occurs while initializing this library.
----------------------	---

**6.28.2.2** `static void activemq::library::ActiveMQCPP::initializeLibrary ( int argc, char ** argv )`  
`[static]`

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

**Parameters**

<i>argc</i>	- the count of arguments passed to this Process.
<i>argv</i>	- the array of string arguments passed to this process.

**Exceptions**

<i>runtime_error</i>	if an error occurs while initializing this library.
----------------------	---

**6.28.2.3** `ActiveMQCPP& activemq::library::ActiveMQCPP::operator= ( const ActiveMQCPP & )` `[protected]`

**6.28.2.4** `static void activemq::library::ActiveMQCPP::shutdownLibrary ( )` `[static]`

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- `src/main/activemq/library/ActiveMQCPP.h`

**6.29 activemq::commands::ActiveMQDestination Class Reference**

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Inheritance diagram for `activemq::commands::ActiveMQDestination`:

**Data Structures**

- struct **DestinationFilter**

## Public Member Functions

- **ActiveMQDestination** ()
- **ActiveMQDestination** (const std::string &physicalName)
- virtual ~**ActiveMQDestination** ()
- virtual **ActiveMQDestination** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual const std::string & **getPhysicalName** () const  
*Fetch this destination's physical name.*
- virtual std::string & **getPhysicalName** ()
- virtual void **setPhysicalName** (const std::string &physicalName)  
*Set this destination's physical name.*
- virtual bool **isAdvisory** () const
- virtual void **setAdvisory** (bool advisory)
- virtual bool **isConsumerAdvisory** () const
- virtual bool **isProducerAdvisory** () const
- virtual bool **isConnectionAdvisory** () const
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool exclusive)
- virtual bool **isOrdered** () const
- virtual void **setOrdered** (bool ordered)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &orderedTarget)
- virtual **cms::Destination::DestinationType** **getDestinationType** () const =0  
*Returns the Type of Destination that this object represents.*
- virtual bool **isTemporary** () const  
*Returns true if a temporary Destination.*
- virtual bool **isTopic** () const  
*Returns true if a Topic Destination.*
- virtual bool **isQueue** () const  
*Returns true if a Queue Destination.*
- virtual bool **isComposite** () const  
*Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.*

- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** \* **getCMSDestination** () const

### Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)  
*Create a temporary name from the clientId.*
- static std::string **getClientId** (const **ActiveMQDestination** \*destination)  
*From a temporary destination find the clientId of the Connection that created it.*
- static **Pointer**< **ActiveMQDestination** > **createDestination** (int type, const std::string &name)  
*Creates a Destination given the String Name to use and a Type.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQDESTINATION** = 0

### Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties** **options**

### Static Protected Attributes

- static const std::string **ADVISORY\_PREFIX**  
*prefix for Advisory message destinations*
- static const std::string **CONSUMER\_ADVISORY\_PREFIX**  
*prefix for consumer advisory destinations*
- static const std::string **PRODUCER\_ADVISORY\_PREFIX**  
*prefix for producer advisory destinations*
- static const std::string **CONNECTION\_ADVISORY\_PREFIX**  
*prefix for connection advisory destinations*
- static const std::string **DEFAULT\_ORDERED\_TARGET**  
*The default target for ordered destinations.*
- static const std::string **TEMP\_PREFIX**
- static const std::string **TEMP\_POSTFIX**
- static const std::string **COMPOSITE\_SEPARATOR**
- static const std::string **QUEUE\_QUALIFIED\_PREFIX**

- static const std::string **TOPIC\_QUALIFIED\_PREFIX**
- static const std::string **TEMP\_QUEUE\_QUALIFIED\_PREFIX**
- static const std::string **TEMP\_TOPIC\_QUALIFIED\_PREFIX**

## 6.29.1 Constructor & Destructor Documentation

6.29.1.1 `activemq::commands::ActiveMQDestination::ActiveMQDestination ( )`

6.29.1.2 `activemq::commands::ActiveMQDestination::ActiveMQDestination ( const std::string & physicalName )`

6.29.1.3 `virtual activemq::commands::ActiveMQDestination::~~ActiveMQDestination ( )`  
[inline, virtual]

## 6.29.2 Member Function Documentation

6.29.2.1 `virtual ActiveMQDestination* activemq::commands::ActiveMQDestination::cloneDataStructure ( )`  
`const` [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1628).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 454), `activemq::commands::ActiveMQTempDestination` (p. 548), `activemq::commands::ActiveMQTempQueue` (p. 575), `activemq::commands::ActiveMQTempTopic` (p. 604), and `activemq::commands::ActiveMQTopic` (p. 661).

6.29.2.2 `virtual void activemq::commands::ActiveMQDestination::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

### Returns

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1629).

Reimplemented in `activemq::commands::ActiveMQQueue` (p. 455), `activemq::commands::ActiveMQTempDestination` (p. 549), `activemq::commands::ActiveMQTempQueue` (p. 576), `activemq::commands::ActiveMQTempTopic` (p. 604), and `activemq::commands::ActiveMQTopic` (p. 662).

Referenced by `activemq::commands::ActiveMQTempDestination::copyDataStructure()`.

6.29.2.3 `static Pointer<ActiveMQDestination> activemq::commands::ActiveMQDestination::createDestination ( int type, const std::string & name ) [static]`

Creates a Destination given the String Name to use and a Type.

#### Parameters

<i>type</i>	- The Type of Destination to Create
<i>name</i>	- The Name to use in the creation of the Destination

#### Returns

Pointer to a new **ActiveMQDestination** (p. 293) instance.

6.29.2.4 `static std::string activemq::commands::ActiveMQDestination::createTemporaryName ( const std::string & clientId ) [inline, static]`

Create a temporary name from the clientId.

#### Parameters

<i>clientId</i>	
-----------------	--

#### Returns

6.29.2.5 `virtual bool activemq::commands::ActiveMQDestination::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 455), **activemq::commands::ActiveMQTempDestination** (p. 549), **activemq::commands::ActiveMQTempQueue** (p. 576), **activemq::commands::ActiveMQTempTopic** (p. 605), and **activemq::commands::ActiveMQTopic** (p. 662).

Referenced by **activemq::commands::ActiveMQTopic::equals()**, and **activemq::commands::ActiveMQTempDestination::equals()**.

6.29.2.6 `static std::string activemq::commands::ActiveMQDestination::getClientId ( const ActiveMQDestination * destination ) [static]`

From a temporary destination find the clientId of the Connection that created it.

#### Parameters

<i>destination</i>
--------------------

#### Returns

the clientId or null if not a temporary destination

6.29.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination ( ) const [inline, virtual]`

#### Returns

the **cms::Destination** (p. 1688) interface pointer that the objects that derive from this class implement.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 455), **activemq::commands::ActiveMQTempQueue** (p. 577), **activemq::commands::ActiveMQTempTopic** (p. 605), and **activemq::commands::ActiveMQTopic** (p. 662).

6.29.2.8 `virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType ( ) const [virtual]`

Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1631).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 456), **activemq::commands::ActiveMQTempQueue** (p. 550), **activemq::commands::ActiveMQTempQueue** (p. 577), **activemq::commands::ActiveMQTempTopic** (p. 605), and **activemq::commands::ActiveMQTopic** (p. 663).

6.29.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQDestination::getDestinationType ( ) const [pure virtual]`

Returns the Type of Destination that this object represents.

#### Returns

int type qualifier.



Implemented in **activemq::commands::ActiveMQQueue** (p. 456), **activemq::commands::ActiveMQTempQueue** (p. 577), **activemq::commands::ActiveMQTempTopic** (p. 606), and **activemq::commands::ActiveMQTopic** (p. 663).

**6.29.2.10** `const activemq::util::ActiveMQProperties&  
activemq::commands::ActiveMQDestination::getOptions ( ) const [inline]`

#### Returns

a reference (const) to the options properties for this Destination.

**6.29.2.11** `virtual std::string activemq::commands::ActiveMQDestination::getOrderedTarget ( )  
const [inline, virtual]`

#### Returns

Returns the orderedTarget.

**6.29.2.12** `virtual const std::string& activemq::commands::ActiveMQDestination::getPhysicalName  
( ) const [inline, virtual]`

Fetch this destination's physical name.

#### Returns

const string containing the name

**6.29.2.13** `virtual std::string& activemq::commands::ActiveMQDestination::getPhysicalName ( )  
[inline, virtual]`

**6.29.2.14** `virtual bool activemq::commands::ActiveMQDestination::isAdvisory ( ) const  
[inline, virtual]`

#### Returns

Returns the advisory.

**6.29.2.15** `virtual bool activemq::commands::ActiveMQDestination::isComposite ( ) const  
[inline, virtual]`

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

#### Returns

true if this destination represents a collection of child destinations.

6.29.2.16 `virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory ( )`  
`const [inline, virtual]`

#### Returns

true if this is a destination for Connection advisories

6.29.2.17 `virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory ( )`  
`const [inline, virtual]`

#### Returns

true if this is a destination for Consumer advisories

6.29.2.18 `virtual bool activemq::commands::ActiveMQDestination::isExclusive ( ) const`  
`[inline, virtual]`

#### Returns

Returns the exclusive.

6.29.2.19 `virtual bool activemq::commands::ActiveMQDestination::isOrdered ( ) const`  
`[inline, virtual]`

#### Returns

Returns the ordered.

6.29.2.20 `virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory ( )`  
`const [inline, virtual]`

#### Returns

true if this is a destination for Producer advisories

6.29.2.21 `virtual bool activemq::commands::ActiveMQDestination::isQueue ( ) const`  
`[inline, virtual]`

Returns true if a Queue Destination.

#### Returns

true/false

6.29.2.22 `virtual bool activemq::commands::ActiveMQDestination::isTemporary ( ) const`  
`[inline, virtual]`

Returns true if a temporary Destination.

#### Returns

true/false

References `cms::Destination::TEMPORARY_QUEUE`, and `cms::Destination::TEMPORARY_TOPIC`.

6.29.2.23 `virtual bool activemq::commands::ActiveMQDestination::isTopic ( ) const`  
`[inline, virtual]`

Returns true if a Topic Destination.

#### Returns

true/false

References `cms::Destination::TEMPORARY_TOPIC`, and `cms::Destination::TOPIC`.

6.29.2.24 `virtual bool activemq::commands::ActiveMQDestination::isWildcard ( ) const`  
`[inline, virtual]`

#### Returns

true if the destination matches multiple possible destinations

6.29.2.25 `virtual void activemq::commands::ActiveMQDestination::setAdvisory ( bool advisory )`  
`[inline, virtual]`

#### Parameters

<i>advisory</i>	The advisory to set.
-----------------	----------------------

6.29.2.26 `virtual void activemq::commands::ActiveMQDestination::setExclusive ( bool exclusive )`  
`[inline, virtual]`

#### Parameters

<i>exclusive</i>	The exclusive to set.
------------------	-----------------------

6.29.2.27 `virtual void activemq::commands::ActiveMQDestination::setOrdered ( bool ordered )`  
`[inline, virtual]`

#### Parameters

<i>ordered</i>	The ordered to set.
----------------	---------------------

6.29.2.28 `virtual void activemq::commands::ActiveMQDestination::setOrderedTarget ( const std::string & orderedTarget )` `[inline, virtual]`

#### Parameters

<i>orderedTarget</i>	The orderedTarget to set.
----------------------	---------------------------

6.29.2.29 `virtual void activemq::commands::ActiveMQDestination::setPhysicalName ( const std::string & physicalName )` `[virtual]`

Set this destination's physical name.

#### Returns

const string containing the name

6.29.2.30 `virtual std::string activemq::commands::ActiveMQDestination::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 457), **activemq::commands::ActiveMQTempQueue** (p. 550), **activemq::commands::ActiveMQTempQueue** (p. 578), **activemq::commands::ActiveMQTempTopic** (p. 606), and **activemq::commands::ActiveMQTopic** (p. 663).

### 6.29.3 Field Documentation

6.29.3.1 `bool activemq::commands::ActiveMQDestination::advisory`  
`[protected]`

**6.29.3.2** `const std::string activemq::commands::ActiveMQDestination::ADVISORY_PREFIX` `[static, protected]`

prefix for Advisory message destinations

**6.29.3.3** `const std::string activemq::commands::ActiveMQDestination::COMPOSITE_SEPARATOR` `[static, protected]`

**6.29.3.4** `const std::string activemq::commands::ActiveMQDestination::CONNECTION_ADVISORY_PREFIX` `[static, protected]`

prefix for connection advisory destinations

**6.29.3.5** `const std::string activemq::commands::ActiveMQDestination::CONSUMER_ADVISORY_PREFIX` `[static, protected]`

prefix for consumer advisory destinations

**6.29.3.6** `const std::string activemq::commands::ActiveMQDestination::DEFAULT_ORDERED_TARGET` `[static, protected]`

The default target for ordered destinations.

**6.29.3.7** `bool activemq::commands::ActiveMQDestination::exclusive` `[protected]`

**6.29.3.8** `const unsigned char activemq::commands::ActiveMQDestination::ID_ACTIVEMQDESTINATION = 0` `[static]`

**6.29.3.9** `util::ActiveMQProperties activemq::commands::ActiveMQDestination::options` `[protected]`

**6.29.3.10** `bool activemq::commands::ActiveMQDestination::ordered` `[protected]`

**6.29.3.11** `std::string activemq::commands::ActiveMQDestination::orderedTarget` `[protected]`

**6.29.3.12** `std::string activemq::commands::ActiveMQDestination::physicalName` `[protected]`

**6.29.3.13** `const std::string activemq::commands::ActiveMQDestination::PRODUCER_ADVISORY_PREFIX` `[static, protected]`

prefix for producer advisory destinations

- 6.29.3.14 `const std::string activemq::commands::ActiveMQDestination::QUEUE_ - QUALIFIED_PREFIX` `[static, protected]`
- 6.29.3.15 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - POSTFIX` `[static, protected]`
- 6.29.3.16 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - PREFIX` `[static, protected]`
- 6.29.3.17 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - QUEUE_QUALIFIED_PREFIX` `[static, protected]`
- 6.29.3.18 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - TOPIC_QUALIFIED_PREFIX` `[static, protected]`
- 6.29.3.19 `const std::string activemq::commands::ActiveMQDestination::TOPIC_ - QUALIFIED_PREFIX` `[static, protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

## 6.30 `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 304).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestina
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`:

### Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*

## 6.30

activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller

### Class Reference

305

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.30.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 304).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.30.2 Constructor & Destructor Documentation

6.30.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller**  
( ) [*inline*]

6.30.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller**  
( ) [*inline, virtual*]

### 6.30.3 Member Function Documentation

6.30.3.1 **virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** )  
[*virtual*]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 462), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 552), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 580), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 608), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 666).

```
6.30.3.2 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 462), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 552), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 580), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 609), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 666).

```
6.30.3.3 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.30

**activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

### Class Reference

307

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 463), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 553), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 581), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 609), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 666).

6.30.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 463), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 554), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 581), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 610), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 667).

6.30.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 464), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 554), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 582), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 610), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 667).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h`

## 6.31 **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 308).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**:

**Public Member Functions**

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*

## 6.31

activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller

### Class Reference

309

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 308).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.31.2 Constructor & Destructor Documentation

6.31.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller**  
( ) [*inline*]

6.31.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller**  
( ) [*inline, virtual*]

### 6.31.3 Member Function Documentation

6.31.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** )  
[*virtual*]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 466), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 584), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 616), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 674).

```
6.31.3.2 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 466), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 584), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 617), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 674).

```
6.31.3.3 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### 6.31

**activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**

#### Class Reference

311

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 467), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 557), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 585), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 617), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 674).

6.31.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 467), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 557), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 585), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 618), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 675).

6.31.3.5 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 468), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 558), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 586), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 618), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 675).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h`

## 6.32 **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 312).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**:

**Public Member Functions**

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*

## 6.32

activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller

### Class Reference

313

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.32.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p.312).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.32.2 Constructor & Destructor Documentation

6.32.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller**  
( ) [*inline*]

6.32.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller**  
( ) [*inline, virtual*]

### 6.32.3 Member Function Documentation

6.32.3.1 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** )  
[*virtual*]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 470), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 560), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 588), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 612), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 670).

```
6.32.3.2 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 470), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 560), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 588), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 613), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 670).

```
6.32.3.3 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.32

**activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller****Class Reference****315****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 471), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 561), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 589), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 613), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 670).

```
6.32.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 471), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 561), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 589), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 614), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 671).

```
6.32.3.5 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 472), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 562), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 590), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 614), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 671).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h`

### 6.33 **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 316).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**:

**Public Member Functions**

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*

### 6.33

activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller

#### Class Reference

317

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.33.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p.316).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.33.2 Constructor & Destructor Documentation

6.33.2.1 activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller  
( ) [inline]

6.33.2.2 virtual activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller  
( ) [inline, virtual]

#### 6.33.3 Member Function Documentation

6.33.3.1 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseMarshal  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure,  
**decaf::io::DataOutputStream** \* dataOut ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 474), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 563), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 592), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 620), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 678).

```
6.33.3.2 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 474), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 564), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 592), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 621), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 678).

```
6.33.3.3 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### 6.33

**activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**

**Class Reference**

**319**

**Returns**

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 475), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 564), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 593), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 621), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 678).

**6.33.3.4** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 475), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 565), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 593), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 622), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 679).

**6.33.3.5** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 476), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 566), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 594), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 622), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 679).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h`

## 6.34 **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 320).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**:

**Public Member Functions**

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*

## 6.34

activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller

### Class Reference

321

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.34.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 320).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.34.2 Constructor & Destructor Documentation

6.34.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller  
( ) [inline]

6.34.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller  
( ) [inline, virtual]

### 6.34.3 Member Function Documentation

6.34.3.1 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseMarshal  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure,  
**decaf::io::DataOutputStream** \* dataOut ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 478), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 567), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 596), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 624), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 686).

```
6.34.3.2 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 478), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 568), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 596), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 625), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 686).

```
6.34.3.3 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.34

**activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

**Class Reference**

**323**

**Returns**

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 479), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 568), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 597), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 625), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 686).

6.34.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 479), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 569), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 597), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 626), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 687).

6.34.3.5 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 480), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 569), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 598), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 626), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 687).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h`

## 6.35 **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 324).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller**:

**Public Member Functions**

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*

## 6.35

activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller

### Class Reference

325

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.35.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 324).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.35.2 Constructor & Destructor Documentation

6.35.2.1 activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller  
( ) [inline]

6.35.2.2 virtual activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller  
( ) [inline, virtual]

### 6.35.3 Member Function Documentation

6.35.3.1 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseMarshal  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure,  
**decaf::io::DataOutputStream** \* dataOut ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 482), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 571), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 600), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 628), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 682).

```
6.35.3.2 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 482), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 572), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 600), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 629), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 682).

```
6.35.3.3 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.35

**activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller****Class Reference****327****Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 483), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 572), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 601), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 629), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 682).

```
6.35.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 483), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 573), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 601), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 630), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 683).

```
6.35.3.5 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 484), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 573), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 602), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 630), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 683).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h`

**6.36 activemq::exceptions::ActiveMQException Class Reference**

```
#include <src/main/activemq/exceptions/ActiveMQException.h>
```

Inheritance diagram for **activemq::exceptions::ActiveMQException**:

**Public Member Functions**

- **ActiveMQException** () throw ()  
*Default Constructor.*
- **ActiveMQException** (const **ActiveMQException** &ex) throw ()  
*Copy Constructor.*
- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()  
*Copy Constructor.*
- **ActiveMQException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** \* **clone** () const  
*Clones this exception.*
- virtual **cms::CMSException** **convertToCMSException** () const  
*Converts this exception to a new CMSException.*

### 6.36.1 Constructor & Destructor Documentation

6.36.1.1 `activemq::exceptions::ActiveMQException::ActiveMQException ( ) throw ()`

Default Constructor.

6.36.1.2 `activemq::exceptions::ActiveMQException::ActiveMQException ( const ActiveMQException & ex ) throw ()`

Copy Constructor.

#### Parameters

<i>ex</i>	The Exception whose internal data is copied into this instance.
-----------	---

6.36.1.3 `activemq::exceptions::ActiveMQException::ActiveMQException ( const decaf::lang::Exception & ex ) throw ()`

Copy Constructor.

#### Parameters

<i>ex</i>	The Exception whose internal data is copied into this instance.
-----------	---

6.36.1.4 `activemq::exceptions::ActiveMQException::ActiveMQException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

#### Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report.
<i>...</i>	The list of primitives that are formatted into the message.

6.36.1.5 `virtual activemq::exceptions::ActiveMQException::~~ActiveMQException ( ) throw ()`  
[virtual]

### 6.36.2 Member Function Documentation

6.36.2.1 `virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone ( )`  
`const [virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

Copy of this Exception object

Reimplemented from `decaf::lang::Exception` (p. 1797).

Reimplemented in `activemq::exceptions::BrokerException` (p. 828).

6.36.2.2 `virtual cms::CMSException activemq::exceptions::ActiveMQException::convertToCMSException ( )const` `[virtual]`

Converts this exception to a new CMSException.

#### Returns

a CMSException with the data from this exception

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/ActiveMQException.h`

## 6.37 `activemq::commands::ActiveMQMapMessage` Class Reference

```
#include <src/main/activemq/commands/ActiveMQMapMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQMapMessage`:

#### Public Member Functions

- `ActiveMQMapMessage ( )`
- `virtual ~ActiveMQMapMessage ( )`
- `virtual unsigned char getDataStructureType ( ) const`  
*Get the unique identifier that this object and its own Marshaler share.*
- `virtual bool isMarshalAware ( ) const`  
*Determine if this object is aware of marshaling and should have its before and after marshaling methods called.*
- `virtual ActiveMQMapMessage * cloneDataStructure ( ) const`



*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat) throw ( **dcf::io::IOException** )  
*Perform any processing needed before an marshal.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw ( **cms::CMSException** )  
*Clears out the body of the message.*
- virtual **cms::MapMessage** \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual std::vector< std::string > **getMapNames** () const throw ( **cms::CMSException** )  
*Returns an Enumeration of all the names in the MapMessage object.*
- virtual bool **itemExists** (const std::string &name) const throw ( **cms::CMSException** )  
*Indicates whether an item exists in this MapMessage object.*
- virtual bool **getBoolean** (const std::string &name) const throw ( **cms::MessageFormatException**, **cms::CMSException** )  
*Returns the Boolean value of the Specified name.*
- virtual void **setBoolean** (const std::string &name, bool value) throw ( **cms::MessageNotWriteableException**, **cms::CMSException** )  
*Sets a boolean value with the specified name into the Map.*
- virtual unsigned char **getByte** (const std::string &name) const throw ( **cms::MessageFormatException**, **cms::CMSException** )  
*Returns the Byte value of the Specified name.*
- virtual void **setByte** (const std::string &name, unsigned char value) throw ( **cms::MessageNotWriteableException**, **cms::CMSException** )  
*Sets a Byte value with the specified name into the Map.*
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const throw ( **cms::MessageFormatException**, **cms::CMSException** )  
*Returns the Bytes value of the Specified name.*
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value) throw ( **cms::MessageNotWriteableException**, **cms::CMSException** )  
*Sets a Bytes value with the specified name into the Map.*
- virtual char **getChar** (const std::string &name) const throw ( **cms::MessageFormatException**, **cms::CMSException** )  
*Returns the Char value of the Specified name.*

- virtual void **setChar** (const std::string &name, char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Char value with the specified name into the Map.*
- virtual double **getDouble** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Double value of the Specified name.*
- virtual void **setDouble** (const std::string &name, double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Double value with the specified name into the Map.*
- virtual float **getFloat** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Float value of the Specified name.*
- virtual void **setFloat** (const std::string &name, float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Float value with the specified name into the Map.*
- virtual int **getInt** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Int value of the Specified name.*
- virtual void **setInt** (const std::string &name, int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Int value with the specified name into the Map.*
- virtual long long **getLong** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Long value of the Specified name.*
- virtual void **setLong** (const std::string &name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Long value with the specified name into the Map.*
- virtual short **getShort** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the Short value of the Specified name.*
- virtual void **setShort** (const std::string &name, short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a Short value with the specified name into the Map.*
- virtual std::string **getString** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Returns the String value of the Specified name.*
- virtual void **setString** (const std::string &name, const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a String value with the specified name into the Map.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQMAPMESSAGE** = 25

## Protected Member Functions

- **util::PrimitiveMap & getMap ()** throw ( decaf::lang::exceptions::NullPointerException )  
*Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.*
- const **util::PrimitiveMap & getMap ()** const throw ( decaf::lang::exceptions::NullPointerException )
- virtual void **checkMapIsUnmarshalled ()** const throw ( decaf::lang::exceptions::NullPointerException )  
*Performs the unmarshal on the Map if needed, otherwise just returns.*

## 6.37.1 Constructor & Destructor Documentation

6.37.1.1 **activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ( )**

6.37.1.2 **virtual activemq::commands::ActiveMQMapMessage::~~ActiveMQMapMessage ( )**  
 [virtual]

## 6.37.2 Member Function Documentation

6.37.2.1 **virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal ( wireformat::WireFormat \* wireFormat )** throw ( decaf::io::IOException )  
 [virtual]

Perform any processing needed before an marshal.

### Parameters

<i>wireFormat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

Implements **activemq::wireformat::MarshalAware** (p. 2445).

6.37.2.2 **virtual void activemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled ( )** const throw ( decaf::lang::exceptions::NullPointerException )  
 [protected, virtual]

Performs the unmarshal on the Map if needed, otherwise just returns.

6.37.2.3 **virtual void activemq::commands::ActiveMQMapMessage::clearBody ( )** throw ( cms::CMSException ) [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 398).

6.37.2.4 `virtual cms::MapMessage* activemq::commands::ActiveMQMapMessage::clone ( ) const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements **cms::Message** (p. 2498).

6.37.2.5 `virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2480).

6.37.2.6 `virtual void activemq::commands::ActiveMQMapMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2481).

6.37.2.7 `virtual bool activemq::commands::ActiveMQMapMessage::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 399).

6.37.2.8 virtual bool activemq::commands::ActiveMQMapMessage::getBoolean ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Boolean value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2434).

6.37.2.9 virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Byte value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2434).

6.37.2.10 virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]

Returns the Bytes value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
---------------------	--

<i>MessageFormatException</i>	- if this type conversion is invalid.
-------------------------------	---------------------------------------

Implements **cms::MapMessage** (p.2435).

6.37.2.11 `virtual char activemq::commands::ActiveMQMapMessage::getChar ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSEException ) [virtual]`

Returns the Char value of the Specified name.

#### Parameters

<i>name</i>	name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p.2435).

6.37.2.12 `virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Message** (p. 2483).

6.37.2.13 `virtual double activemq::commands::ActiveMQMapMessage::getDouble ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSEException ) [virtual]`

Returns the Double value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSEException</i>	- if the operation fails due to an internal error.
----------------------	--

<i>MessageFormatException</i>	- if this type conversion is invalid.
-------------------------------	---------------------------------------

Implements **cms::MapMessage** (p. 2435).

6.37.2.14 `virtual float activemq::commands::ActiveMQMapMessage::getFloat ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Float value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2436).

6.37.2.15 `virtual int activemq::commands::ActiveMQMapMessage::getInt ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Int value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2436).

6.37.2.16 `virtual long long activemq::commands::ActiveMQMapMessage::getLong ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSException ) [virtual]`

Returns the Long value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p.2437).

6.37.2.17 `util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap ( )  
throw ( decaf::lang::exceptions::NullPointerException ) [protected]`

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

**Returns**

reference to a PrimitiveMap.

6.37.2.18 `const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap  
( ) const throw ( decaf::lang::exceptions::NullPointerException )  
[protected]`

6.37.2.19 `virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames ( )  
const throw ( cms::CMSEException ) [virtual]`

Returns an Enumeration of all the names in the MapMessage object.

**Returns**

STL Vector of String values, each of which is the name of an item in the MapMessage

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
----------------------	--

Implements **cms::MapMessage** (p.2437).

6.37.2.20 `virtual short activemq::commands::ActiveMQMapMessage::getShort ( const  
std::string & name ) const throw ( cms::MessageFormatException,  
cms::CMSEException ) [virtual]`

Returns the Short value of the Specified name.



**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2437).

```
6.37.2.21 virtual std::string activemq::commands::ActiveMQMapMessage::getString (
    const std::string & name ) const throw ( cms::MessageFormatException,
    cms::CMSException ) [virtual]
```

Returns the String value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageFormatException</i>	- if this type conversion is invalid.

Implements **cms::MapMessage** (p. 2438).

```
6.37.2.22 virtual bool activemq::commands::ActiveMQMapMessage::isMarshalAware ( ) const
    [inline, virtual]
```

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

**Returns**

true if aware of marshaling

Reimplemented from **activemq::commands::Message** (p. 2486).

```
6.37.2.23 virtual bool activemq::commands::ActiveMQMapMessage::itemExists ( const
    std::string & name ) const throw ( cms::CMSException ) [virtual]
```

Indicates whether an item exists in this MapMessage object.

**Parameters**

<i>name</i>	String name of the Object in question
-------------	---------------------------------------

**Returns**

boolean value indicating if the name is in the map

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
----------------------	--

Implements **cms::MapMessage** (p.2438).

```
6.37.2.24 virtual void activemq::commands::ActiveMQMapMessage::setBoolean ( const
std::string & name, bool value ) throw ( cms::MessageNotWriteableException,
cms::CMSEException ) [virtual]
```

Sets a boolean value with the specified name into the Map.

**Parameters**

<i>name</i>	the name of the boolean
<i>value</i>	the boolean value to set in the Map

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p.2439).

```
6.37.2.25 virtual void activemq::commands::ActiveMQMapMessage::setByte ( const std::string
& name, unsigned char value ) throw ( cms::MessageNotWriteableException,
cms::CMSEException ) [virtual]
```

Sets a Byte value with the specified name into the Map.

**Parameters**

<i>name</i>	the name of the Byte
<i>value</i>	the Byte value to set in the Map

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p.2439).

```
6.37.2.26 virtual void activemq::commands::ActiveMQMapMessage::setBytes ( const
std::string & name, const std::vector< unsigned char > & value ) throw
( cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Sets a Bytes value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Bytes
<i>value</i>	The Bytes value to set in the Map

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2439).

```
6.37.2.27 virtual void activemq::commands::ActiveMQMapMessage::setChar ( const std::string
& name, char value ) throw ( cms::MessageNotWriteableException,
cms::CMSException ) [virtual]
```

Sets a Char value with the specified name into the Map.

#### Parameters

<i>name</i>	the name of the Char
<i>value</i>	the Char value to set in the Map

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2440).

```
6.37.2.28 virtual void activemq::commands::ActiveMQMapMessage::setDouble
( const std::string & name, double value ) throw (
cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Sets a Double value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Double
<i>value</i>	The Double value to set in the Map

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p.2440).

```
6.37.2.29 virtual void activemq::commands::ActiveMQMapMessage::setFloat ( const std::string
& name, float value ) throw ( cms::MessageNotWriteableException,
cms::CMSEException ) [virtual]
```

Sets a Float value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Float
<i>value</i>	The Float value to set in the Map

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p.2441).

```
6.37.2.30 virtual void activemq::commands::ActiveMQMapMessage::setInt ( const std::string
& name, int value ) throw ( cms::MessageNotWriteableException,
cms::CMSEException ) [virtual]
```

Sets a Int value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Int
<i>value</i>	The Int value to set in the Map

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p.2441).

6.37.2.31 `virtual void activemq::commands::ActiveMQMapMessage::setLong ( const std::string & name, long long value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Long value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Long
<i>value</i>	The Long value to set in the Map

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2442).

6.37.2.32 `virtual void activemq::commands::ActiveMQMapMessage::setShort ( const std::string & name, short value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a Short value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Short
<i>value</i>	The Short value to set in the Map

#### Exceptions

<i>CMSException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2442).

6.37.2.33 `virtual void activemq::commands::ActiveMQMapMessage::setString ( const std::string & name, const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Sets a String value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the String
<i>value</i>	The String value to set in the Map

**Exceptions**

<i>CMSEException</i>	- if the operation fails due to an internal error.
<i>MessageNotWriteableException</i>	- if the <b>Message</b> (p. 2475) is in Read-only Mode.

Implements **cms::MapMessage** (p. 2442).

```
6.37.2.34 virtual std::string activemq::commands::ActiveMQMapMessage::toString ( ) const
           [virtual]
```

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2490).

**6.37.3 Field Documentation**

```
6.37.3.1 const unsigned char activemq::commands::ActiveMQMapMessage::ID_ -
          ACTIVEMQMAPMESSAGE = 25 [static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMapMessage.h**

## 6.38 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 344).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMess
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller:

**Public Member Functions**

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const

## 6.38

### activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller Class Reference 345

*Creates a new instance of this marshalable type.*

- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.38.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 344).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.38.2 Constructor & Destructor Documentation

6.38.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller**  
( ) [inline]

6.38.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller**  
( ) [inline, virtual]

#### 6.38.3 Member Function Documentation

6.38.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.38.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.38.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )
    [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2658).

```
6.38.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.38

**activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**  
**Class Reference** **347**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

6.38.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

6.38.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2660).

```
6.38.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2661).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMapMessageMarshaller.h**

## 6.39 **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 348).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMess
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**:

**Public Member Functions**

- **ActiveMQMapMessageMarshaller ()**

## 6.39

### activemq:wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller Class Reference 349

- virtual `~ActiveMQMapMessageMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

#### 6.39.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 348).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.39.2 Constructor & Destructor Documentation

6.39.2.1 `activemq:wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]`

6.39.2.2 `virtual activemq:wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ( ) [inline, virtual]`

#### 6.39.3 Member Function Documentation

6.39.3.1 `virtual commands::DataStructure* activemq:wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.39.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.39.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )
    [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2671).

```
6.39.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

### 6.39

## activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller

### Class Reference

351

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

6.39.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

6.39.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673).

```
6.39.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2674).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQMapMessageMarshaller.h**

## 6.40 **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 352).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMess
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**:

**Public Member Functions**

- **ActiveMQMapMessageMarshaller ()**

## 6.40

### activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller Class Reference 353

- virtual `~ActiveMQMapMessageMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

#### 6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 352).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.40.2 Constructor & Destructor Documentation

6.40.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]`

6.40.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ( ) [inline, virtual]`

#### 6.40.3 Member Function Documentation

6.40.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.40.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.40.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2667).

```
6.40.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.40

### activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller Class Reference 355

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

6.40.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

6.40.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

```
6.40.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMapMessageMarshaller.h**

## 6.41 **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 356).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMess
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller**:

**Public Member Functions**

- **ActiveMQMapMessageMarshaller ()**

## 6.41

### activemq:wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller Class Reference 357

- virtual `~ActiveMQMapMessageMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

#### 6.41.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 356).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.41.2 Constructor & Destructor Documentation

6.41.2.1 `activemq:wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]`

6.41.2.2 `virtual activemq:wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ( ) [inline, virtual]`

#### 6.41.3 Member Function Documentation

6.41.3.1 `virtual commands::DataStructure* activemq:wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.41.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.41.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )
    [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2654).

```
6.41.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.41

### activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller Class Reference 359

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.41.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.41.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

6.41.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h`

## 6.42 **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 360).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMess
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**:

**Public Member Functions**

- **ActiveMQMapMessageMarshaller ()**

## 6.42

### activemq:wireformat:openwire:marshal:v2:ActiveMQMapMessageMarshaller Class Reference 361

- virtual `~ActiveMQMapMessageMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

#### 6.42.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 360).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.42.2 Constructor & Destructor Documentation

6.42.2.1 `activemq:wireformat:openwire:marshal:v2:ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]`

6.42.2.2 `virtual activemq:wireformat:openwire:marshal:v2:ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ( ) [inline, virtual]`

#### 6.42.3 Member Function Documentation

6.42.3.1 `virtual commands::DataStructure* activemq:wireformat:openwire:marshal:v2:ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.42.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.42.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )
    [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

```
6.42.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.42

**activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**  
**Class Reference** **363**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

6.42.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

6.42.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.42.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2665).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMapMessageMarshaller.h**

## 6.43 activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 364).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMess
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller**:

**Public Member Functions**

- **ActiveMQMapMessageMarshaller ()**

## 6.43

### activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller Class Reference 365

- virtual `~ActiveMQMapMessageMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

#### 6.43.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 364).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.43.2 Constructor & Destructor Documentation

6.43.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller ( ) [inline]`

6.43.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller ( ) [inline, virtual]`

#### 6.43.3 Member Function Documentation

6.43.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.43.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.43.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

```
6.43.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

## 6.43

### activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller Class Reference 367

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

6.43.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

6.43.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

```
6.43.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2678).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQMapMessageMarshaller.h**

**6.44 activemq::commands::ActiveMQMessage Class Reference**

```
#include <src/main/activemq/commands/ActiveMQMessage.h>
```

Inheritance diagram for **activemq::commands::ActiveMQMessage**:

**Public Member Functions**

- **ActiveMQMessage** ()
- virtual **~ActiveMQMessage** ()
- virtual unsigned char **getDataStructureType** () const  
Get the unique identifier that this object and its own Marshaler share.
- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this objects members, overwriting any existing data.*

- virtual **ActiveMQMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQMESSAGE** = 23

### 6.44.1 Constructor & Destructor Documentation

6.44.1.1 **activemq::commands::ActiveMQMessage::ActiveMQMessage** ( )

6.44.1.2 **virtual activemq::commands::ActiveMQMessage::~~ActiveMQMessage** ( )  
 [inline, virtual]

### 6.44.2 Member Function Documentation

6.44.2.1 **virtual cms::Message\*** **activemq::commands::ActiveMQMessage::clone** ( ) const  
 [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements **cms::Message** (p. 2498).

6.44.2.2 **virtual ActiveMQMessage\*** **activemq::commands::ActiveMQMessage::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2480).

6.44.2.3 `virtual void activemq::commands::ActiveMQMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2481).

6.44.2.4 `virtual bool activemq::commands::ActiveMQMessage::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 399).

6.44.2.5 `virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Message** (p. 2483).

6.44.2.6 `virtual std::string activemq::commands::ActiveMQMessage::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.



## 6.45 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class Reference 371

---

Reimplemented from **activemq::commands::Message** (p. 2490).

### 6.44.3 Field Documentation

6.44.3.1 `const unsigned char activemq::commands::ActiveMQMessage::ID_ -  
ACTIVEMQMESSAGE = 23` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessage.h`

## 6.45 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 371).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.45.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 371).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.45.2 Constructor & Destructor Documentation

6.45.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.45.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.45.3 Member Function Documentation

6.45.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.45.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.45.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseMarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
(p. 2658).

6.45.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
(p. 2659).

6.45.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal1  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

```
6.45.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2660).

```
6.45.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

## 6.46 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller

### Class Reference

375

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2661).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMessageMarshaller.h**

## 6.46 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 375).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**:

#### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 375).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.46.2 Constructor & Destructor Documentation

6.46.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.46.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.46.3 Member Function Documentation

6.46.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.46.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.46.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
`[virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2671).

**6.46.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
`[virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

**6.46.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` `[virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

```
6.46.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673).

```
6.46.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.47 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller

### Class Reference

379

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2674).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQMessageMarshaller.h**

## 6.47 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 379).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**:

#### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual **~ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.47.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 379).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.47.2 Constructor & Destructor Documentation

6.47.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.47.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.47.3 Member Function Documentation

6.47.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.47.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.47 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller

### Class Reference 381

6.47.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` `[virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2667).

6.47.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` `[virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

6.47.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` `[virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

```
6.47.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

```
6.47.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

## 6.48 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller

### Class Reference

383

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQMessageMarshaller.h**

## 6.48 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 383).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.48.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 383).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.48.2 Constructor & Destructor Documentation

6.48.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.48.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.48.3 Member Function Documentation

6.48.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.48.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.48.3.3** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
`[virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2654).

**6.48.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
`[virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

**6.48.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` `[virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

```
6.48.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

```
6.48.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQMessageMarshaller.h**

## 6.49 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 387).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**:

#### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.49.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 387).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.49.2 Constructor & Destructor Documentation

6.49.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.49.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.49.3 Member Function Documentation

6.49.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.49.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.49.3.3** virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseMarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )  
 [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**  
 (p. 2663).

**6.49.3.4** virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**  
 (p. 2663).

**6.49.3.5** virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal1  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.49.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.49.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2665).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMessageMarshaller.h**

## 6.50 activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 391).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller:

### Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.50.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 391).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.50.2 Constructor & Destructor Documentation

6.50.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller**  
( ) [inline]

6.50.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller**  
( ) [inline, virtual]

### 6.50.3 Member Function Documentation

6.50.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.50.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.50.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseMarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**  
(p. 2676).

6.50.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::looseUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**  
(p. 2676).

6.50.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal1  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

```
6.50.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

```
6.50.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.51 activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference 395

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2678).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQMessageMarshaller.h**

## 6.51 activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference

```
#include <src/main/activemq/commands/ActiveMQMessageTemplate.h>
```

Inheritance diagram for **activemq::commands::ActiveMQMessageTemplate< T >**:

### Public Member Functions

- **ActiveMQMessageTemplate** ()
- virtual ~**ActiveMQMessageTemplate** ()
- virtual void **acknowledge** () const throw ( cms::IllegalStateException, cms::CMSException )  
*Acknowledges all consumed messages of the session of this consumed message.*
- virtual void **onSend** ()  
*Resets the **Message** (p. 2475) to a Read-Only state.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **clearProperties** () throw ( cms::CMSException )  
*Clears the message properties.*
- virtual std::vector< std::string > **getPropertyNames** () const throw ( cms::CMSException )  
*Retrieves the property names.*

- virtual bool **propertyExists** (const std::string &name) const throw ( cms::CMSEException )  
*Indicates whether or not a given property exists.*
- virtual bool **getBooleanProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a double property.*
- virtual float **getFloatProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a float property.*
- virtual int **getIntProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a int property.*
- virtual long long **getLongProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a long property.*
- virtual short **getShortProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a short property.*
- virtual std::string **getStringProperty** (const std::string &name) const throw ( cms::MessageFormatException, cms::CMSEException )  
*Gets a string property.*
- virtual void **setBooleanProperty** (const std::string &name, bool value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a boolean property.*
- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a byte property.*
- virtual void **setDoubleProperty** (const std::string &name, double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a double property.*
- virtual void **setFloatProperty** (const std::string &name, float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a float property.*
- virtual void **setIntProperty** (const std::string &name, int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a int property.*
- virtual void **setLongProperty** (const std::string &name, long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Sets a long property.*

- virtual void **setShortProperty** (const std::string &name, short value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets a short property.*
- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets a string property.*
- virtual std::string **getCMSCorrelationID** () const throw ( cms::CMSException )  
*Get the Correlation Id for this message.*
- virtual void **setCMSCorrelationID** (const std::string &correlationId) throw ( cms::CMSException )  
*Sets the Correlation Id used by this message.*
- virtual int **getCMSDeliveryMode** () const throw ( cms::CMSException )  
*Gets the DeliveryMode for this message.*
- virtual void **setCMSDeliveryMode** (int mode) throw ( cms::CMSException )  
*Sets the DeliveryMode for this message.*
- virtual const cms::Destination \* **getCMSDestination** () const throw ( cms::CMSException )  
*Gets the Destination for this **Message** (p. 2475), returns a.*
- virtual void **setCMSDestination** (const cms::Destination \*destination) throw ( cms::CMSException )  
*Sets the Destination for this message.*
- virtual long long **getCMSExpiration** () const throw ( cms::CMSException )  
*Gets the Expiration Time for this **Message** (p. 2475).*
- virtual void **setCMSExpiration** (long long expireTime) throw ( cms::CMSException )  
*Sets the Expiration Time for this message.*
- virtual std::string **getCMSMessageID** () const throw ( cms::CMSException )  
*Gets the CMS **Message** (p. 2475) Id for this **Message** (p. 2475).*
- virtual void **setCMSMessageID** (const std::string &id AMQCPP\_UNUSED) throw ( cms::CMSException )  
*Sets the CMS **Message** (p. 2475) Id for this message.*
- virtual int **getCMSPriority** () const throw ( cms::CMSException )  
*Gets the Priority Value for this **Message** (p. 2475).*
- virtual void **setCMSPriority** (int priority) throw ( cms::CMSException )  
*Sets the Priority Value for this message.*
- virtual bool **getCMSRedelivered** () const throw ( cms::CMSException )  
*Gets the Redelivered Flag for this **Message** (p. 2475).*
- virtual void **setCMSRedelivered** (bool redelivered AMQCPP\_UNUSED) throw ( cms::CMSException )  
*Sets the Redelivered Flag for this message.*
- virtual const cms::Destination \* **getCMSReplyTo** () const throw ( cms::CMSException )  
*Gets the CMS Reply To Address for this **Message** (p. 2475).*
- virtual void **setCMSReplyTo** (const cms::Destination \*destination) throw ( cms::CMSException )

*Sets the CMS Reply To Address for this message.*

- virtual long long **getCMSTimestamp** () const throw ( cms::CMSEException )

*Gets the Time Stamp for this **Message** (p. 2475).*

- virtual void **setCMSTimestamp** (long long timeStamp) throw ( cms::CMSEException )

*Sets the Time Stamp for this message.*

- virtual std::string **getCMSType** () const throw ( cms::CMSEException )

*Gets the CMS **Message** (p. 2475) Type for this **Message** (p. 2475).*

- virtual void **setCMSType** (const std::string &type) throw ( cms::CMSEException )

*Sets the CMS **Message** (p. 2475) Type for this message.*

## Protected Member Functions

- void **failIfWriteOnlyBody** () const
- void **failIfReadOnlyBody** () const
- void **failIfReadOnlyProperties** () const

```
template<typename T> class activemq::commands::ActiveMQMessageTemplate< T >
```

### 6.51.1 Constructor & Destructor Documentation

```
6.51.1.1 template<typename T> activemq::commands::ActiveMQMessageTemplate<
T>::ActiveMQMessageTemplate ( ) [inline]
```

```
6.51.1.2 template<typename T> virtual activemq::commands::ActiveMQMessageTemplate<
T>::~~ActiveMQMessageTemplate ( ) [inline, virtual]
```

### 6.51.2 Member Function Documentation

```
6.51.2.1 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::acknowledge ( ) const throw ( cms::IllegalStateException,
cms::CMSEException ) [inline, virtual]
```

Acknowledges all consumed messages of the session of this consumed message.

```
6.51.2.2 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::clearBody ( ) throw ( cms::CMSEException ) [inline, virtual]
```

Clears out the body of the message.

This does not clear the headers or properties.

## 6.51 activemq::commands::ActiveMQMessageTemplate< T > Class Template

### Reference

399

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 205), **activemq::commands::ActiveMQMapMessage** (p. 333), **activemq::commands::ActiveMQStreamMessage** (p. 509), and **activemq::commands::ActiveMQTextMessage** (p. 632).

```
6.51.2.3 template<typename T> virtual void
    activemq::commands::ActiveMQMessageTemplate< T
    >::clearProperties ( ) throw ( cms::CMSException ) [inline,
    virtual]
```

Clears the message properties.

Does not clear the body or header values.

```
6.51.2.4 template<typename T> virtual bool
    activemq::commands::ActiveMQMessageTemplate< T
    >::equals ( const DataStructure * value ) const [inline, virtual]
```

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 2481).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 175), **activemq::commands::ActiveMQBytesMessage** (p. 206), **activemq::commands::ActiveMQMapMessage** (p. 334), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 415), **activemq::commands::ActiveMQStreamMessage** (p. 510), and **activemq::commands::ActiveMQTextMessage** (p. 633).

```
6.51.2.5 template<typename T> void activemq::commands::ActiveMQMessageTemplate<
    T >::failIfReadOnlyBody ( ) const [inline, protected]
```

```
6.51.2.6 template<typename T> void activemq::commands::ActiveMQMessageTemplate<
    T >::failIfReadOnlyProperties ( ) const [inline, protected]
```

```
6.51.2.7 template<typename T> void activemq::commands::ActiveMQMessageTemplate<
    T >::failIfWriteOnlyBody ( ) const [inline, protected]
```

```
6.51.2.8 template<typename T> virtual bool
    activemq::commands::ActiveMQMessageTemplate< T
    >::getBooleanProperty ( const std::string & name ) const throw (
    cms::MessageFormatException, cms::CMSException ) [inline,
    virtual]
```

Gets a boolean property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**Exceptions**

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.9  template<typename T> virtual unsigned char
activemq::commands::ActiveMQMessageTemplate< T >::getBytesProperty
( const std::string & name ) const throw ( cms::MessageFormatException,
cms::CMSEException ) [inline, virtual]
```

Gets a byte property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**Exceptions**

<i>CMSEException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.10 template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSCorrelationID ( ) const throw ( cms::CMSEException ) [inline,
virtual]
```

Get the Correlation Id for this message.

**Returns**

string representation of the correlation Id

**Exceptions**

<i>CMSEException</i>	
----------------------	--

6.51.2.11 `template<typename T> virtual int  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSDeliveryMode ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the DeliveryMode for this message.

#### Returns

DeliveryMode enumerated value.

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.51.2.12 `template<typename T> virtual const cms::Destination*  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSDestination ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the Destination for this **Message** (p. 2475), returns a.

#### Returns

Destination object

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.51.2.13 `template<typename T> virtual long long  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSExpiration ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the Expiration Time for this **Message** (p. 2475).

#### Returns

time value

#### Exceptions

<i>CMSEException</i>	
----------------------	--

```
6.51.2.14  template<typename T> virtual std::string
           activemq::commands::ActiveMQMessageTemplate< T
           >::getCMSMessageID ( ) const throw ( cms::CMSEException ) [inline,
           virtual]
```

Gets the CMS **Message** (p. 2475) Id for this **Message** (p. 2475).

#### Returns

time value

#### Exceptions

<i>CMSEException</i>
----------------------

```
6.51.2.15  template<typename T> virtual int
           activemq::commands::ActiveMQMessageTemplate< T
           >::getCMSPriority ( ) const throw ( cms::CMSEException ) [inline,
           virtual]
```

Gets the Priority Value for this **Message** (p. 2475).

#### Returns

priority value

#### Exceptions

<i>CMSEException</i>
----------------------

```
6.51.2.16  template<typename T> virtual bool
           activemq::commands::ActiveMQMessageTemplate< T
           >::getCMSRedelivered ( ) const throw ( cms::CMSEException ) [inline,
           virtual]
```

Gets the Redelivered Flag for this **Message** (p. 2475).

#### Returns

redelivered value

#### Exceptions

<i>CMSEException</i>
----------------------



6.51.2.17 `template<typename T> virtual const cms::Destination*  
activemq::commands::ActiveMQMessageTemplate< T >::getCMSReplyTo  
( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the CMS Reply To Address for this **Message** (p. 2475).

#### Returns

Reply To Value

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.51.2.18 `template<typename T> virtual long long  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSTimestamp ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the Time Stamp for this **Message** (p. 2475).

#### Returns

time stamp value

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.51.2.19 `template<typename T> virtual std::string  
activemq::commands::ActiveMQMessageTemplate< T  
>::getCMSType ( ) const throw ( cms::CMSEException ) [inline,  
virtual]`

Gets the CMS **Message** (p. 2475) Type for this **Message** (p. 2475).

#### Returns

type value

#### Exceptions

<i>CMSEException</i>	
----------------------	--

```

6.51.2.20 template<typename T> virtual double
activemq::commands::ActiveMQMessageTemplate< T
>::getDoubleProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]

```

Gets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```

6.51.2.21 template<typename T> virtual float
activemq::commands::ActiveMQMessageTemplate< T
>::getFloatProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]

```

Gets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.22 template<typename T> virtual int
activemq::commands::ActiveMQMessageTemplate< T
>::getIntProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]
```

Gets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.23 template<typename T> virtual long long
activemq::commands::ActiveMQMessageTemplate< T
>::getLongProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]
```

Gets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.24 template<typename T> virtual std::vector<std::string>
activemq::commands::ActiveMQMessageTemplate< T
>::getPropertyNames ( ) const throw ( cms::CMSException ) [inline,
virtual]
```

Retrieves the property names.

#### Returns

The complete set of property names currently in this message.

```
6.51.2.25 template<typename T> virtual short
activemq::commands::ActiveMQMessageTemplate< T
>::getShortProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]
```

Gets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<i>CMSException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

```
6.51.2.26 template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getStringProperty ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [inline,
virtual]
```

Gets a string property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

## Exceptions

<i>CMSException</i>	if the property does not exist.
<i>MessageFormatException</i>	- if this type conversion is invalid.

6.51.2.27 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::onSend( ) [inline, virtual]`

Resets the **Message** (p. 2475) to a Read-Only state.

Reimplemented from **activemq::commands::Message** (p. 2487).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 207), and **activemq::commands::ActiveMQStreamMessage** (p. 511).

6.51.2.28 `template<typename T> virtual bool  
activemq::commands::ActiveMQMessageTemplate< T  
>::propertyExists( const std::string & name ) const throw ( cms::CMSException  
) [inline, virtual]`

Indicates whether or not a given property exists.

## Parameters

<i>name</i>	The name of the property to look up.
-------------	--------------------------------------

## Returns

True if the property exists in this message.

6.51.2.29 `template<typename T> virtual void  
activemq::commands::ActiveMQMessageTemplate< T  
>::setBooleanProperty( const std::string & name, bool value ) throw ( cms::MessageNotWriteableException, cms::CMSException )  
[inline, virtual]`

Sets a boolean property.

## Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

## Exceptions

<i>CMSException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.30  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setByteProperty ( const std::string & name, unsigned char value ) throw
            ( cms::MessageNotWriteableException, cms::CMSEException )
            [inline, virtual]

```

Sets a byte property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.31  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSCorrelationID ( const std::string & correlationId ) throw (
            cms::CMSEException ) [inline, virtual]

```

Sets the Correlation Id used by this message.

#### Parameters

<i>correlationId</i>	- String representing the correlation id.
----------------------	---

#### Exceptions

<i>CMSEException</i>	
----------------------	--

```

6.51.2.32  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSDeliveryMode ( int mode ) throw ( cms::CMSEException )
            [inline, virtual]

```

Sets the DeliveryMode for this message.

#### Parameters

<i>mode</i>	- DeliveryMode enumerated value.
-------------	----------------------------------

#### Exceptions

<i>CMSEException</i>	
----------------------	--

```
6.51.2.33  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSDestination ( const cms::Destination * destination ) throw (
            cms::CMSException ) [inline, virtual]
```

Sets the Destination for this message.

#### Parameters

<i>destination</i>	- Destination Object
--------------------	----------------------

#### Exceptions

<i>CMSException</i>	
---------------------	--

```
6.51.2.34  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSExpiration ( long long expireTime ) throw ( cms::CMSException )
            [inline, virtual]
```

Sets the Expiration Time for this message.

#### Parameters

<i>expireTime</i>	- time value
-------------------	--------------

#### Exceptions

<i>CMSException</i>	
---------------------	--

```
6.51.2.35  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSMessageID ( const std::string &id AMQCPP_UNUSED ) throw (
            cms::CMSException ) [inline, virtual]
```

Sets the CMS **Message** (p. 2475) Id for this message.

#### Parameters

<i>id</i>	- time value
-----------	--------------

#### Exceptions

<i>CMSException</i>	
---------------------	--

```
6.51.2.36  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSPriority ( int priority ) throw ( cms::CMSEException )  [inline,
           virtual]
```

Sets the Priority Value for this message.

#### Parameters

<i>priority</i>	- priority value for this message
-----------------	-----------------------------------

#### Exceptions

<i>CMSEException</i>
----------------------

```
6.51.2.37  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSRedelivered ( bool redelivered AMQCPP_UNUSED ) throw (
           cms::CMSEException )  [inline, virtual]
```

Sets the Redelivered Flag for this message.

#### Parameters

<i>redelivered</i>	- boolean redelivered value
--------------------	-----------------------------

#### Exceptions

<i>CMSEException</i>
----------------------

```
6.51.2.38  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSReplyTo ( const cms::Destination * destination ) throw (
           cms::CMSEException )  [inline, virtual]
```

Sets the CMS Reply To Address for this message.

#### Parameters

<i>destination</i>	Pointer to the CMS Destination that is the Reply-To value.
--------------------	--

#### Exceptions

<i>CMSEException</i>
----------------------



```
6.51.2.39  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSTimestamp ( long long timeStamp ) throw ( cms::CMSEException )
            [inline, virtual]
```

Sets the Time Stamp for this message.

#### Parameters

<i>timeStamp</i>	- integer time stamp value
------------------	----------------------------

#### Exceptions

<i>CMSEException</i>	
----------------------	--

```
6.51.2.40  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setCMSType ( const std::string & type ) throw ( cms::CMSEException )
            [inline, virtual]
```

Sets the CMS **Message** (p. 2475) Type for this message.

#### Parameters

<i>type</i>	- message type value string
-------------	-----------------------------

#### Exceptions

<i>CMSEException</i>	
----------------------	--

```
6.51.2.41  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setDoubleProperty ( const std::string & name, double value ) throw
            ( cms::MessageNotWriteableException, cms::CMSEException )
            [inline, virtual]
```

Sets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.42  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setFloatProperty ( const std::string & name, float value ) throw (
            cms::MessageNotWriteableException, cms::CMSException )
            [inline, virtual]

```

Sets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.43  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setIntProperty ( const std::string & name, int value ) throw (
            cms::MessageNotWriteableException, cms::CMSException )
            [inline, virtual]

```

Sets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<i>CMSException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```

6.51.2.44  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setLongProperty ( const std::string & name, long long value ) throw
            ( cms::MessageNotWriteableException, cms::CMSException )
            [inline, virtual]

```

Sets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

<i>value</i>	The value for the named property.
--------------	-----------------------------------

**Exceptions**

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```
6.51.2.45 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setShortProperty ( const std::string & name, short value ) throw (
cms::MessageNotWriteableException, cms::CMSEException )
[inline, virtual]
```

Sets a short property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**Exceptions**

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

```
6.51.2.46 template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setStringProperty ( const std::string & name, const std::string & value )
throw ( cms::MessageNotWriteableException, cms::CMSEException )
[inline, virtual]
```

Sets a string property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

**Exceptions**

<i>CMSEException</i>	- if the name is an empty string.
<i>MessageNotWriteableException</i>	- if properties are read-only

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessageTemplate.h**

## 6.52 activemq::commands::ActiveMQObjectMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQObjectMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

### Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual **~ActiveMQObjectMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQObjectMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQOBJECTMESSAGE** = 26

### 6.52.1 Constructor & Destructor Documentation

6.52.1.1 **activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage** ( )

6.52.1.2 **virtual activemq::commands::ActiveMQObjectMessage::~~ActiveMQObjectMessage** ( )  
[inline, virtual]

### 6.52.2 Member Function Documentation

6.52.2.1 `virtual cms::Message* activemq::commands::ActiveMQObjectMessage::clone ( )`  
`const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

Implements **cms::Message** (p. 2498).

6.52.2.2 `virtual ActiveMQObjectMessage* activemq::commands::ActiveMQObjectMessage::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2480).

6.52.2.3 `virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2481).

6.52.2.4 `virtual bool activemq::commands::ActiveMQObjectMessage::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 399).

**6.52.2.5** `virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Message** (p. 2483).

**6.52.2.6** `virtual std::string activemq::commands::ActiveMQObjectMessage::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2490).

### 6.52.3 Field Documentation

**6.52.3.1** `const unsigned char activemq::commands::ActiveMQObjectMessage::ID_ - ACTIVEMQOBJECTMESSAGE = 26 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQObjectMessage.h**

## 6.53 **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 416).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectM
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**:

- **ActiveMQObjectMessageMarshaller** ()
- virtual **~ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.53.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 416).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.53.2 Constructor & Destructor Documentation

6.53.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller** ( ) [*inline*]

6.53.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller** ( ) [*inline, virtual*]

### 6.53.3 Member Function Documentation

6.53.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.53.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.53.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2658).



## 6.53 ac-

activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller

### Class Reference

419

```
6.53.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

```
6.53.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

6.53.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2660).

6.53.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2661).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h`

6.54 ac-

activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller

Class Reference

6.54 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 421).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.54.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 421).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.54.2 Constructor & Destructor Documentation

6.54.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller ( ) [inline]`

6.54.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller ( ) [inline, virtual]`

## 6.54.3 Member Function Documentation

6.54.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.54.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.54.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.54 ac-

**activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**  
**Class Reference** **423**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2671).

**6.54.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

**6.54.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

6.54.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673).

6.54.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2674).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h`

6.55 ac-

activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller

Class Reference

6.55 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 425).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.55.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 425).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.55.2 Constructor & Destructor Documentation

6.55.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller ( ) [inline]`

6.55.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller ( ) [inline, virtual]`

## 6.55.3 Member Function Documentation

6.55.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.55.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.55.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## 6.55 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller

### Class Reference 427

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2667).

**6.55.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

**6.55.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

6.55.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

6.55.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h`

6.56 ac-

activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller

Class Reference

6.56 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller

429

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 429).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.56.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 429).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.56.2 Constructor & Destructor Documentation

6.56.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller ( ) [inline]`

6.56.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller ( ) [inline, virtual]`

## 6.56.3 Member Function Documentation

6.56.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.56.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.56.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## 6.56 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller

### Class Reference 431

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2654).

**6.56.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

**6.56.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.56.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

6.56.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQObjectMessageMarshaller.h**

6.57 ac-

activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller

Class Reference

6.57 ~~activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller~~ 433

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 433).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.57.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 433).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.57.2 Constructor & Destructor Documentation

6.57.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller ( ) [inline]`

6.57.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller ( ) [inline, virtual]`

## 6.57.3 Member Function Documentation

6.57.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.57.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.57.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



**6.57 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**  
**Class Reference** **435**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

**6.57.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

**6.57.3.5** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

6.57.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

6.57.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2665).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQObjectMessageMarshaller.h**

6.58 ac-

activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller

Class Reference

6.58 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 437).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller:

### Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.58.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 437).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.58.2 Constructor & Destructor Documentation

6.58.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller ( ) [inline]`

6.58.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::~~ActiveMQObjectMessageMarshaller ( ) [inline, virtual]`

## 6.58.3 Member Function Documentation

6.58.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.58.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.58.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**6.58 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**  
**Class Reference** **439**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

**6.58.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

**6.58.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

6.58.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

6.58.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2678).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMarshaller.h`

## 6.59 activemq::core::ActiveMQProducer Class Reference

```
#include <src/main/activemq/core/ActiveMQProducer.h>
```

Inheritance diagram for activemq::core::ActiveMQProducer:

### Public Member Functions

- **ActiveMQProducer** (**ActiveMQSession** \*session, const **Pointer**< **commands::ProducerId** > &producerId, const **Pointer**< **commands::ActiveMQDestination** > &destination, long long sendTimeout)  
*Constructor, creates an instance of an **ActiveMQProducer** (p. 441).*
- virtual ~**ActiveMQProducer** ()
- virtual void **close** () throw ( cms::CMSException )  
*Closes the Consumer.*
- virtual void **send** (**cms::Message** \*message) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (**cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **setDeliveryMode** (int mode) throw ( cms::CMSException )  
*Sets the delivery mode for this Producer.*
- virtual int **getDeliveryMode** () const throw ( cms::CMSException )  
*Gets the delivery mode for this Producer.*
- virtual void **setDisableMessageID** (bool value) throw ( cms::CMSException )  
*Sets if Message Ids are disabled for this Producer.*
- virtual bool **getDisableMessageID** () const throw ( cms::CMSException )  
*Gets if Message Ids are disabled for this Producer.*

- virtual void **setDisableMessageTimeStamp** (bool value) throw ( cms::CMSEException )  
*Sets if Message Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () const throw ( cms::CMSEException )  
*Gets if Message Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority) throw ( cms::CMSEException )  
*Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const throw ( cms::CMSEException )  
*Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time) throw ( cms::CMSEException )  
*Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const throw ( cms::CMSEException )  
*Gets the Time to Live that this producer sends messages with.*
- virtual void **setSendTimeout** (long long time) throw ( cms::CMSEException )  
*Sets the Send Timeout that this Producers sends messages with.*
- virtual long long **getSendTimeout** () const throw ( cms::CMSEException )  
*Gets the Send Timeout that this producer sends messages with.*
- bool **isClosed** () const
- const **Pointer**< **commands::ProducerInfo** > & **getProducerInfo** () const  
*Retries this object ProducerInfo pointer.*
- const **Pointer**< **commands::ProducerId** > & **getProducerId** () const  
*Retries this object ProducerId or NULL if closed.*
- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)  
*Handles the work of Processing a ProducerAck Command from the Broker.*

### 6.59.1 Constructor & Destructor Documentation

- 6.59.1.1 **activemq::core::ActiveMQProducer::ActiveMQProducer** ( **ActiveMQSession** \* session, const **Pointer**< **commands::ProducerId** > & producerId, const **Pointer**< **commands::ActiveMQDestination** > & destination, long long sendTimeout )

Constructor, creates an instance of an **ActiveMQProducer** (p. 441).

#### Parameters

<i>session</i>	The Session which is the parent of this Producer.
<i>producerId</i>	Pointer to a ProducerId object which identifies this producer.
<i>destination</i>	The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.
<i>sendTime-out</i>	The configured send timeout for this Producer.



6.59.1.2 `virtual activemq::core::ActiveMQProducer::~~ActiveMQProducer ( ) [virtual]`

## 6.59.2 Member Function Documentation

6.59.2.1 `virtual void activemq::core::ActiveMQProducer::close ( ) throw ( cms::CMSEException ) [virtual]`

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Closeable** (p. 1120).

6.59.2.2 `virtual int activemq::core::ActiveMQProducer::getDeliveryMode ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the delivery mode for this Producer.

### Returns

The DeliveryMode

Implements **cms::MessageProducer** (p. 2683).

6.59.2.3 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageID ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

### Returns

a boolean indicating state enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p. 2683).

6.59.2.4 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageTimeStamp ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

### Returns

boolean indicating state of enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2684).

6.59.2.5 `virtual int activemq::core::ActiveMQProducer::getPriority ( ) const throw ( cms::CMSException ) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

#### Returns

int based priority level

Implements **cms::MessageProducer** (p. 2684).

6.59.2.6 `const Pointer<commands::ProducerId>& activemq::core::ActiveMQProducer::getProducerId ( ) const [inline]`

Retries this object ProducerId or NULL if closed.

#### Returns

ProducerId Reference

6.59.2.7 `const Pointer<commands::ProducerInfo>& activemq::core::ActiveMQProducer::getProducerInfo ( ) const [inline]`

Retries this object ProducerInfo pointer.

#### Returns

ProducerInfo Reference

6.59.2.8 `virtual long long activemq::core::ActiveMQProducer::getSendTimeout ( ) const throw ( cms::CMSException ) [inline, virtual]`

Gets the Send Timeout that this producer sends messages with.

#### Returns

The default send timeout value in milliseconds.

6.59.2.9 `virtual long long activemq::core::ActiveMQProducer::getTimeToLive ( ) const throw ( cms::CMSException ) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

#### Returns

The default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2684).

6.59.2.10 `bool activemq::core::ActiveMQProducer::isClosed ( ) const` `[inline]`

#### Returns

true if this Producer has been closed.

6.59.2.11 `virtual void activemq::core::ActiveMQProducer::onProducerAck ( const commands::ProducerAck & ack )` `[virtual]`

Handles the work of Processing a ProducerAck Command from the Broker.

#### Parameters

<i>ack</i>	- The ProducerAck message received from the Broker.
------------	---

6.59.2.12 `virtual void activemq::core::ActiveMQProducer::send ( const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )` `[virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2685).

6.59.2.13 `virtual void activemq::core::ActiveMQProducer::send ( cms::Message * message ) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

#### Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2686).

6.59.2.14 `virtual void activemq::core::ActiveMQProducer::send ( cms::Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<i>CMSException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2685).

6.59.2.15 `virtual void activemq::core::ActiveMQProducer::send ( const cms::Destination * destination, cms::Message * message ) throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException ) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2687).

6.59.2.16 `virtual void activemq::core::ActiveMQProducer::setDeliveryMode ( int mode ) throw ( cms::CMSEException ) [inline, virtual]`

Sets the delivery mode for this Producer.

#### Parameters

<i>mode</i>	- The DeliveryMode to use for Message sends.
-------------	--

Implements **cms::MessageProducer** (p. 2687).

6.59.2.17 `virtual void activemq::core::ActiveMQProducer::setDisableMessageID ( bool value ) throw ( cms::CMSEException ) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

#### Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Implements **cms::MessageProducer** (p. 2688).

6.59.2.18 `virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp ( bool value ) throw ( cms::CMSEException ) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

#### Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

Implements **cms::MessageProducer** (p. 2688).

6.59.2.19 `virtual void activemq::core::ActiveMQProducer::setPriority ( int priority ) throw ( cms::CMSEException ) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

#### Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

Implements **cms::MessageProducer** (p. 2688).

6.59.2.20 `virtual void activemq::core::ActiveMQProducer::setSendTimeout ( long long time ) throw ( cms::CMSEException ) [inline, virtual]`

Sets the Send Timeout that this Producers sends messages with.

#### Parameters

<i>time</i>	The new default send timeout value in milliseconds.
-------------	---

6.59.2.21 `virtual void activemq::core::ActiveMQProducer::setTimeToLive ( long long time ) throw ( cms::CMSEException ) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

#### Parameters

<i>time</i>	The new default time to live value in milliseconds.
-------------	---

Implements **cms::MessageProducer** (p. 2689).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQProducer.h**

## 6.60 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3072) object.

```
#include <src/main/activemq/util/ActiveMQProperties.h>
```

Inheritance diagram for activemq::util::ActiveMQProperties:

### Public Member Functions

- **ActiveMQProperties** ()
- virtual **~ActiveMQProperties** ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual const **decaf::util::Properties** & **getProperties** () const
- virtual void **setProperty** (**decaf::util::Properties** &props)
- virtual bool **isEmpty** () const  
*Returns true if the properties object is empty.*
- virtual const char \* **getProperty** (const std::string &name) const  
*Looks up the value for the given property.*
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const  
*Looks up the value for the given property.*
- virtual void **setProperty** (const std::string &name, const std::string &value)  
*Sets the value for a given property.*
- virtual bool **hasProperty** (const std::string &name) const  
*Check to see if the Property exists in the set.*
- virtual void **remove** (const std::string &name)  
*Removes the property with the given name.*
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const  
*Method that serializes the contents of the property map to an array.*
- virtual void **copy** (const CMSProperties \*source)  
*Copies the contents of the given properties object to this one.*
- virtual CMSProperties \* **clone** () const  
*Clones this object.*
- virtual void **clear** ()  
*Clears all properties from the map.*
- virtual std::string **toString** () const  
*Formats the contents of the Properties Object into a string that can be logged, etc.*

### 6.60.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3072) object.

#### Since

2.0

### 6.60.2 Constructor & Destructor Documentation

6.60.2.1 `activemq::util::ActiveMQProperties::ActiveMQProperties ( )`

6.60.2.2 `virtual activemq::util::ActiveMQProperties::~~ActiveMQProperties ( )` [virtual]

### 6.60.3 Member Function Documentation

6.60.3.1 `virtual void activemq::util::ActiveMQProperties::clear ( )` [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 1136).

6.60.3.2 `virtual CMSProperties* activemq::util::ActiveMQProperties::clone ( )` const [virtual]

Clones this object.

#### Returns

a replica of this object.

Implements **cms::CMSProperties** (p. 1136).

6.60.3.3 `virtual void activemq::util::ActiveMQProperties::copy ( const CMSProperties * source )` [virtual]

Copies the contents of the given properties object to this one.

#### Parameters

<i>source</i>	The source properties object.
---------------	-------------------------------

6.60.3.4 `virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ( )` [inline, virtual]



6.60.3.5 `virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ( ) const`  
[inline, virtual]

6.60.3.6 `virtual const char* activemq::util::ActiveMQProperties::getProperty ( const std::string & name ) const` [inline, virtual]

Looks up the value for the given property.

#### Parameters

<i>name</i>	The name of the property to be looked up.
-------------	---

#### Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements **cms::CMSProperties** (p. 1136).

6.60.3.7 `virtual std::string activemq::util::ActiveMQProperties::getProperty ( const std::string & name, const std::string & defaultValue ) const` [inline, virtual]

Looks up the value for the given property.

#### Parameters

<i>name</i>	the name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

#### Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implements **cms::CMSProperties** (p. 1137).

6.60.3.8 `virtual bool activemq::util::ActiveMQProperties::hasProperty ( const std::string & name ) const` [inline, virtual]

Check to see if the Property exists in the set.

#### Parameters

<i>name</i>	- property name to check for in this properties set.
-------------	--

#### Returns

true if property exists, false otherwise.

Implements **cms::CMSProperties** (p. 1137).

6.60.3.9 `virtual bool activemq::util::ActiveMQProperties::isEmpty ( ) const [inline, virtual]`

Returns true if the properties object is empty.

#### Returns

true if empty

Implements **cms::CMSProperties** (p. 1137).

6.60.3.10 `virtual void activemq::util::ActiveMQProperties::remove ( const std::string & name ) [inline, virtual]`

Removes the property with the given name.

#### Parameters

<i>name</i>	the name of the property to remove.
-------------	-------------------------------------

Implements **cms::CMSProperties** (p. 1138).

6.60.3.11 `virtual void activemq::util::ActiveMQProperties::setProperties ( decaf::util::Properties & props ) [inline, virtual]`

6.60.3.12 `virtual void activemq::util::ActiveMQProperties::setProperty ( const std::string & name, const std::string & value ) [inline, virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

#### Parameters

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

Implements **cms::CMSProperties** (p. 1138).

6.60.3.13 `virtual std::vector< std::pair< std::string, std::string > > activemq::util::ActiveMQProperties::toArray ( ) const [inline, virtual]`

Method that serializes the contents of the property map to an array.

#### Returns

list of pairs where the first is the name and the second is the value.

Implements **cms::CMSProperties** (p. 1138).

6.60.3.14 `virtual std::string activemq::util::ActiveMQProperties::toString ( ) const`  
`[inline, virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

### Returns

string value of this object.

Implements **cms::CMSProperties** (p. 1138).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/ActiveMQProperties.h`

## 6.61 activemq::commands::ActiveMQQueue Class Reference

```
#include <src/main/activemq/commands/ActiveMQQueue.h>
```

Inheritance diagram for `activemq::commands::ActiveMQQueue`:

### Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual **~ActiveMQQueue** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.*
- virtual **ActiveMQQueue \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** \* **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual **cms::Destination** \* **clone** () const

*Creates a new instance of this destination type that is a copy of this one, and returns it.*

- virtual void **copy** (const **cms::Destination** &source)  
*Copies the contents of the given Destination object to this one.*
- virtual const **cms::CMSProperties** & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getQueueName** () const throw ( cms::CMSException )  
*Gets the name of this queue.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQUEUE** = 100

### 6.61.1 Constructor & Destructor Documentation

6.61.1.1 **activemq::commands::ActiveMQQueue::ActiveMQQueue** ( )

6.61.1.2 **activemq::commands::ActiveMQQueue::ActiveMQQueue** ( const std::string & *name* )

6.61.1.3 **virtual activemq::commands::ActiveMQQueue::~~ActiveMQQueue** ( ) [inline, virtual]

### 6.61.2 Member Function Documentation

6.61.2.1 **virtual cms::Destination\*** **activemq::commands::ActiveMQQueue::clone** ( ) const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

#### Returns

cloned copy of this object

Implements **cms::Destination** (p. 1690).

6.61.2.2 **virtual ActiveMQQueue\*** **activemq::commands::ActiveMQQueue::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 296).

6.61.2.3 `virtual void activemq::commands::ActiveMQQueue::copy ( const cms::Destination & source ) [inline, virtual]`

Copies the contents of the given Destination object to this one.

#### Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1690).

6.61.2.4 `virtual void activemq::commands::ActiveMQQueue::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 296).

6.61.2.5 `virtual bool activemq::commands::ActiveMQQueue::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 297).

6.61.2.6 `virtual const cms::Destination* activemq::commands::ActiveMQQueue::getCMSDestination ( ) const [inline, virtual]`

#### Returns

the **cms::Destination** (p. 1688) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 298).

```
6.61.2.7 virtual const cms::CMSProperties& activemq::commands::ActiveMQQueue::getCMSProperties ( )  
const [inline, virtual]
```

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

#### Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1690).

```
6.61.2.8 virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType ( ) const [virtual]
```

Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 298).

```
6.61.2.9 virtual cms::Destination::DestinationType  
activemq::commands::ActiveMQQueue::getDestinationType ( ) const [inline,  
virtual]
```

Retrieve the Destination Type for this Destination.

#### Returns

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 298).

References cms::Destination::QUEUE.

```
6.61.2.10 virtual std::string activemq::commands::ActiveMQQueue::getQueueName ( ) const  
throw ( cms::CMSException ) [inline, virtual]
```

Gets the name of this queue.

#### Returns

The queue name.

Implements **cms::Queue** (p. 3094).

6.61.2.11 `virtual std::string activemq::commands::ActiveMQQueue::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 302).

### 6.61.3 Field Documentation

6.61.3.1 `const unsigned char activemq::commands::ActiveMQQueue::ID_-`  
`ACTIVEMQQUEUE = 100 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQQueue.h`

## 6.62 activemq::core::ActiveMQQueueBrowser Class Reference

```
#include <src/main/activemq/core/ActiveMQQueueBrowser.h>
```

Inheritance diagram for `activemq::core::ActiveMQQueueBrowser`:

### Public Member Functions

- **ActiveMQQueueBrowser** (**ActiveMQSession** \*session, const **Pointer**< **commands::ConsumerId** > &consumerId, const **Pointer**< **commands::ActiveMQDestination** > &destination, const std::string &selector, bool dispatchAsync)
- virtual **~ActiveMQQueueBrowser** ()
- virtual const **cms::Queue** \* **getQueue** () const throw ( cms::CMSEException )
- virtual std::string **getMessageSelector** () const throw ( cms::CMSEException )
- virtual **cms::MessageEnumeration** \* **getEnumeration** () throw ( cms::CMSEException )  
*Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.*
- virtual void **close** () throw ( cms::CMSEException )  
*Closes this object and deallocates the appropriate resources.*
- virtual bool **hasMoreMessages** ()  
*Returns true if there are more Message in the Browser that can be retrieved via the nextMessage method.*

- virtual **cms::Message \* nextMessage ()** throw ( cms::CMSEException )  
*Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.*

## Friends

- class **Browser**

## 6.62.1 Constructor & Destructor Documentation

6.62.1.1 **activemq::core::ActiveMQQueueBrowser::ActiveMQQueueBrowser ( ActiveMQSession \* session, const Pointer< commands::ConsumerId > & consumerId, const Pointer< commands::ActiveMQDestination > & destination, const std::string & selector, bool dispatchAsync )**

6.62.1.2 **virtual activemq::core::ActiveMQQueueBrowser::~~ActiveMQQueueBrowser ( )**  
 [virtual]

## 6.62.2 Member Function Documentation

6.62.2.1 **virtual void activemq::core::ActiveMQQueueBrowser::close ( )** throw ( cms::CMSEException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

## Exceptions

<i>CMSEException</i>	- If an error occurs while the resource is being closed.
----------------------	--

Implements **cms::Closeable** (p. 1120).

6.62.2.2 **virtual cms::MessageEnumeration\* activemq::core::ActiveMQQueueBrowser::getEnumeration ( )** throw ( cms::CMSEException ) [virtual]

Gets a pointer to an Enumeration object for browsing the Messages currently in the Queue in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

## Returns

a pointer to a Queue Enumeration, this Pointer is owned by the QueueBrowser and should not be deleted by the client.



### Exceptions

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 3099).

6.62.2.3 `virtual std::string activemq::core::ActiveMQQueueBrowser::getMessageSelector ( )`  
`const throw ( cms::CMSEException )` [virtual]

### Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

### Exceptions

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 3099).

6.62.2.4 `virtual const cms::Queue* activemq::core::ActiveMQQueueBrowser::getQueue ( )`  
`const throw ( cms::CMSEException )` [virtual]

### Returns

the Queue that this browser is listening on.

### Exceptions

<i>CMSEException</i>	if an internal error occurs.
----------------------	------------------------------

Implements **cms::QueueBrowser** (p. 3100).

6.62.2.5 `virtual bool activemq::core::ActiveMQQueueBrowser::hasMoreMessages ( )`  
[virtual]

Returns true if there are more Message in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

### Returns

true if more Message's are available in the Browser.

Implements **cms::MessageEnumeration** (p. 2620).

6.62.2.6 `virtual cms::Message* activemq::core::ActiveMQQueueBrowser::nextMessage ( )`  
`throw ( cms::CMSException ) [virtual]`

Returns the Next Message in the Queue if one is present, if no more Message's are available then an Exception is thrown.

If a Message object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

### Returns

The next Message in the Queue.

### Exceptions

<i>CMSException</i> if no more Message's currently in the Queue.
--

Implements **cms::MessageEnumeration** (p. 2621).

## 6.62.3 Friends And Related Function Documentation

6.62.3.1 `friend class Browser [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQQueueBrowser.h`

## 6.63 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 460).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMa`

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`:

### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.63.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 460).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.63.2 Constructor & Destructor Documentation

6.63.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.63.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.63.3 Member Function Documentation

6.63.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.63.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::getDataStructureType( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.63.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 305).

6.63.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.63 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller

### Class Reference 463

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 306).

6.63.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 306).

6.63.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 307).

```
6.63.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 307).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQQueueMarshaller.h**

## 6.64 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 464).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**:

#### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.64.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 464).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.64.2 Constructor & Destructor Documentation

6.64.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.64.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.64.3 Member Function Documentation

6.64.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.64.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::getDataStructureType( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.64.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 309).

6.64.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



## 6.64 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller

### Class Reference 467

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 310).

6.64.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 310).

6.64.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 311).

```
6.64.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 311).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQQueueMarshaller.h**

## 6.65 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 468).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**:

#### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.65.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 468).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.65.2 Constructor & Destructor Documentation

6.65.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.65.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.65.3 Member Function Documentation

6.65.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.65.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::getDataStructureType( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.65.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 313).

6.65.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.65 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller

### Class Reference 471

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 314).

6.65.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 314).

6.65.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 315).

```
6.65.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 315).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQQueueMarshaller.h**

## 6.66 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 472).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**:

#### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.66.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 472).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.66.2 Constructor & Destructor Documentation

6.66.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.66.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.66.3 Member Function Documentation

6.66.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.66.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::getDataStructureType( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.66.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 317).

6.66.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



## 6.66 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller

### Class Reference 475

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 318).

```
6.66.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 318).

```
6.66.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 319).

```
6.66.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 319).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQQueueMarshaller.h**

## 6.67 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 476).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**:

#### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.67.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 476).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.67.2 Constructor & Destructor Documentation

6.67.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.67.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.67.3 Member Function Documentation

6.67.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.67.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::getDataStructureType  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.67.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseMarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 321).

6.67.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.67 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller

### Class Reference 479

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 322).

6.67.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 322).

6.67.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 323).

```
6.67.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 323).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQQueueMarshaller.h**

## 6.68 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 480).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**:

#### Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.68.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 480).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.68.2 Constructor & Destructor Documentation

6.68.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller**  
( ) [inline]

6.68.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller**  
( ) [inline, virtual]

### 6.68.3 Member Function Documentation

6.68.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.68.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::getDataStructureType( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.68.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 325).

6.68.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



## 6.68 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller

### Class Reference 483

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 326).

6.68.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 326).

6.68.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 327).

```
6.68.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 327).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQQueueMarshaller.h**

## 6.69 activemq::core::ActiveMQSession Class Reference

```
#include <src/main/activemq/core/ActiveMQSession.h>
```

Inheritance diagram for **activemq::core::ActiveMQSession**:

#### Public Member Functions

- **ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > &sessionInfo, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties, **ActiveMQConnection** \*connection)
- virtual **~ActiveMQSession** ()
- void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)  
*Redispatches the given set of unconsumed messages to the consumers.*
- void **start** ()  
*Stops asynchronous message delivery.*
- void **stop** ()  
*Starts asynchronous message delivery.*
- bool **isStarted** () const

*Indicates whether or not the session is currently in the started state.*

- bool **isAutoAcknowledge** () const
- bool **isDupsOkAcknowledge** () const
- bool **isClientAcknowledge** () const
- bool **isIndividualAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)

*Fires the given exception to the exception listener of the connection.*

- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)

*Dispatches a message to a particular consumer.*

- virtual void **close** () throw ( cms::CMSEException )

*Closes this session as well as any active child consumers or producers.*

- virtual void **commit** () throw ( cms::CMSEException )

*Commits all messages done in this transaction and releases any locks currently held.*

- virtual void **rollback** () throw ( cms::CMSEException )

*Rollsback all messages done in this transaction and releases any locks currently held.*

- virtual void **recover** () throw ( cms::CMSEException )

*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*

- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )

*Creates a MessageConsumer for the specified destination.*

- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector) throw ( cms::CMSEException )

*Creates a MessageConsumer for the specified destination, using a message selector.*

- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )

*Creates a MessageConsumer for the specified destination, using a message selector.*

- virtual **cms::MessageConsumer** \* **createDurableConsumer** (const **cms::Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false) throw ( cms::CMSEException )

*Creates a durable subscriber to the specified topic, using a message selector.*

- virtual **cms::MessageProducer** \* **createProducer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )

*Creates a MessageProducer to send messages to the specified destination.*

- virtual **cms::QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::QueueBrowser** \* **createBrowser** (const **cms::Queue** \*queue, const std::string &selector) throw ( cms::CMSEException )

*Creates a new QueueBrowser to peek at Messages on the given Queue.*

- virtual **cms::Queue** \* **createQueue** (const std::string &queueName) throw ( cms::CMSEException )

*Creates a queue identity given a Queue name.*

- virtual **cms::Topic** \* **createTopic** (const std::string &topicName) throw ( cms::CMSEException )  
*Creates a topic identity given a Queue name.*
- virtual **cms::TemporaryQueue** \* **createTemporaryQueue** () throw ( cms::CMSEException )  
*Creates a TemporaryQueue object.*
- virtual **cms::TemporaryTopic** \* **createTemporaryTopic** () throw ( cms::CMSEException )  
*Creates a TemporaryTopic object.*
- virtual **cms::Message** \* **createMessage** () throw ( cms::CMSEException )  
*Creates a new Message.*
- virtual **cms::BytesMessage** \* **createBytesMessage** () throw ( cms::CMSEException )  
*Creates a BytesMessage.*
- virtual **cms::BytesMessage** \* **createBytesMessage** (const unsigned char \*bytes, int bytesSize) throw ( cms::CMSEException )  
*Creates a BytesMessage and sets the pay-load to the passed value.*
- virtual **cms::StreamMessage** \* **createStreamMessage** () throw ( cms::CMSEException )  
*Creates a new StreamMessage.*
- virtual **cms::TextMessage** \* **createTextMessage** () throw ( cms::CMSEException )  
*Creates a new TextMessage.*
- virtual **cms::TextMessage** \* **createTextMessage** (const std::string &text) throw ( cms::CMSEException )  
*Creates a new TextMessage and set the text to the value given.*
- virtual **cms::MapMessage** \* **createMapMessage** () throw ( cms::CMSEException )  
*Creates a new MapMessage.*
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw ( cms::CMSEException )  
*Returns the acknowledgment mode of the session.*
- virtual bool **isTransacted** () const throw ( cms::CMSEException )  
*Gets if the Sessions is a Transacted Session.*
- virtual void **unsubscribe** (const std::string &name) throw ( cms::CMSEException )  
*Unsubscribes a durable subscription that has been created by a client.*
- void **send** (cms::Message \*message, **ActiveMQProducer** \*producer, **util::Usage** \*usage) throw ( cms::CMSEException )  
*Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.*
- **cms::ExceptionListener** \* **getExceptionListener** ()  
*This method gets any registered exception listener of this sessions connection and returns it.*

- const **commands::SessionInfo** & **getSessionInfo** () const  
*Gets the Session Information object for this session, if the session is closed than this method throws an exception.*
- const **commands::SessionId** & **getSessionId** () const  
*Gets the Session Id object for this session, if the session is closed than this method throws an exception.*
- **ActiveMQConnection** \* **getConnection** () const  
*Gets the **ActiveMQConnection** (p. 244) that is associated with this session.*
- long long **getLastDeliveredSequenceId** () const  
*Gets the currently set Last Delivered Sequence Id.*
- void **setLastDeliveredSequenceId** (long long value)  
*Sets the value of the Last Delivered Sequence Id.*
- void **oneway** (**Pointer**< **commands::Command** > command) throw ( activemq::exceptions::ActiveMQException )  
*Sends a oneway message.*
- void **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0) throw ( activemq::exceptions::ActiveMQException )  
*Sends a synchronous request and returns the response from the broker.*
- void **addConsumer** (**ActiveMQConsumer** \*consumer) throw ( activemq::exceptions::ActiveMQException )  
*Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.*
- void **removeConsumer** (const **Pointer**< **commands::ConsumerId** > &consumerId, long long lastDeliveredSequenceId=0) throw ( activemq::exceptions::ActiveMQException )  
*Dispose of a MessageConsumer from this session.*
- void **addProducer** (**ActiveMQProducer** \*consumer) throw ( activemq::exceptions::ActiveMQException )  
*Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.*
- void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw ( activemq::exceptions::ActiveMQException )  
*Dispose of a MessageProducer from this session.*
- void **doStartTransaction** () throw ( exceptions::ActiveMQException )  
*Starts if not already start a Transaction for this Session.*
- **Pointer**< **ActiveMQTransactionContext** > **getTransactionContext** ()  
*Gets the Pointer to this Session's TransactionContext.*
- void **acknowledge** ()  
*Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.*
- void **deliverAcks** ()  
*Request that this Session inform all of its consumers to deliver their pending acks.*
- void **clearMessagesInProgress** ()

*Request that this Session inform all of its consumers to clear all messages that are currently in progress.*

- void **wakeup** ()

*Causes the Session to wakeup its executer and ensure all messages are dispatched.*

- **Pointer**< **commands::ConsumerId** > **getNextConsumerId** ()

*Get the Next available Consumer Id.*

- **Pointer**< **commands::ProducerId** > **getNextProducerId** ()

*Get the Next available Producer Id.*

## Friends

- class **ActiveMQSessionExecutor**

## 6.69.1 Constructor & Destructor Documentation

6.69.1.1 **activemq::core::ActiveMQSession::ActiveMQSession** ( const **Pointer**< **commands::SessionInfo** > & *sessionInfo*, **cms::Session::AcknowledgeMode** *ackMode*, const **decaf::util::Properties** & *properties*, **ActiveMQConnection** \* *connection* )

6.69.1.2 **virtual activemq::core::ActiveMQSession::~~ActiveMQSession** ( ) [virtual]

## 6.69.2 Member Function Documentation

6.69.2.1 **void activemq::core::ActiveMQSession::acknowledge** ( )

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

6.69.2.2 **void activemq::core::ActiveMQSession::addConsumer** ( **ActiveMQConsumer** \* *consumer* ) throw ( **activemq::exceptions::ActiveMQException** )

Adds a MessageConsumer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

### Parameters

<i>consumer</i>	The <b>ActiveMQConsumer</b> (p. 282) instance to add to this session.
-----------------	---

### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.69.2.3 `void activemq::core::ActiveMQSession::addProducer ( ActiveMQProducer * consumer ) throw ( activemq::exceptions::ActiveMQException )`

Adds a MessageProducer to this session registering it with the Connection and store a reference to it so the session can ensure that all resources are closed when the session is closed.

#### Parameters

<i>consumer</i>	The <b>ActiveMQProducer</b> (p. 441) instance to add to this session.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.69.2.4 `void activemq::core::ActiveMQSession::clearMessagesInProgress ( )`

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

6.69.2.5 `virtual void activemq::core::ActiveMQSession::close ( ) throw ( cms::CMSException ) [virtual]`

Closes this session as well as any active child consumers or producers.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3309).

6.69.2.6 `virtual void activemq::core::ActiveMQSession::commit ( ) throw ( cms::CMSException ) [virtual]`

Commits all messages done in this transaction and releases any locks currently held.

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3309).

6.69.2.7 `virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser ( const cms::Queue * queue ) throw ( cms::CMSException ) [virtual]`

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters**

<i>queue</i>	the Queue to browse
--------------	---------------------

**Returns**

New QueueBrowser that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3309).

```
6.69.2.8 virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser
( const cms::Queue * queue, const std::string & selector ) throw (
cms::CMSEException ) [virtual]
```

Creates a new QueueBrowser to peek at Messages on the given Queue.

**Parameters**

<i>queue</i>	the Queue to browse
<i>selector</i>	the Message selector to filter which messages are browsed.

**Returns**

New QueueBrowser that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3310).

```
6.69.2.9 virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage ( ) throw (
cms::CMSEException ) [virtual]
```

Creates a BytesMessage.

**Returns**

a newly created BytesMessage.

**Exceptions**

<i>CMSEException</i>	
----------------------	--



Implements **cms::Session** (p. 3310).

6.69.2.10 virtual **cms::BytesMessage\*** **activemq::core::ActiveMQSession::createBytesMessage** ( const unsigned char \* *bytes*, int *bytesSize* ) throw ( cms::CMSEException )  
[virtual]

Creates a BytesMessage and sets the pay-load to the passed value.

#### Parameters

<i>bytes</i>	- an array of bytes to set in the message
<i>bytesSize</i>	- the size of the bytes array, or number of bytes to use

#### Returns

a newly created BytesMessage.

#### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3311).

6.69.2.11 virtual **cms::MessageConsumer\*** **activemq::core::ActiveMQSession::createConsumer** ( const cms::Destination \* *destination*, const std::string & *selector*, bool *noLocal* ) throw ( cms::CMSEException ) [virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	- The Destination that this consumer receiving messages for.
<i>selector</i>	- The Message Selector string to use for this destination
<i>noLocal</i>	- if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

#### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3312).

```
6.69.2.12 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer ( const
        cms::Destination * destination, const std::string & selector ) throw (
        cms::CMSException ) [virtual]
```

Creates a MessageConsumer for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	- The Destination that this consumer receiving messages for.
<i>selector</i>	- The Message Selector string to use for this destination

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3311).

```
6.69.2.13 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer ( const
        cms::Destination * destination ) throw ( cms::CMSException )
        [virtual]
```

Creates a MessageConsumer for the specified destination.

#### Parameters

<i>destination</i>	- The Destination that this consumer receiving messages for.
--------------------	--

#### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::Session** (p. 3311).

```
6.69.2.14 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer ( const
        cms::Topic * destination, const std::string & name, const std::string & selector,
        bool noLocal = false ) throw ( cms::CMSException ) [virtual]
```

Creates a durable subscriber to the specified topic, using a message selector.

#### Parameters

<i>destination</i>	- the topic to subscribe to
<i>name</i>	- The name used to identify the subscription
<i>selector</i>	- only messages matching the selector are received
<i>noLocal</i>	- if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3313).

6.69.2.15 **virtual cms::MapMessage\*** **activemq::core::ActiveMQSession::createMapMessage**  
( ) throw ( **cms::CMSEException** ) [virtual]

Creates a new MapMessage.

### Returns

a newly created MapMessage.

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3314).

6.69.2.16 **virtual cms::Message\*** **activemq::core::ActiveMQSession::createMessage** ( )  
throw ( **cms::CMSEException** ) [virtual]

Creates a new Message.

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3314).

6.69.2.17 **virtual cms::MessageProducer\*** **activemq::core::ActiveMQSession::createProducer** ( **const**  
**cms::Destination \*** *destination* ) throw ( **cms::CMSEException** )  
[virtual]

Creates a MessageProducer to send messages to the specified destination.

### Parameters

<i>destination</i>	- the Destination to publish on
--------------------	---------------------------------

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3314).

6.69.2.18 `virtual cms::Queue* activemq::core::ActiveMQSession::createQueue ( const  
std::string & queueName ) throw ( cms::CMSEException ) [virtual]`

Creates a queue identity given a Queue name.

#### Parameters

<code>queueName</code>	- the name of the new Queue
------------------------	-----------------------------

#### Exceptions

<code>CMSEException</code>
----------------------------

Implements **cms::Session** (p. 3315).

6.69.2.19 `virtual cms::StreamMessage* ac-  
tivismq::core::ActiveMQSession::createStreamMessage ( )  
throw ( cms::CMSEException ) [virtual]`

Creates a new StreamMessage.

#### Returns

a newly created StreamMessage.

#### Exceptions

<code>CMSEException</code>
----------------------------

Implements **cms::Session** (p. 3315).

6.69.2.20 `virtual cms::TemporaryQueue* ac-  
tivismq::core::ActiveMQSession::createTemporaryQueue ( )  
throw ( cms::CMSEException ) [virtual]`

Creates a TemporaryQueue object.

#### Exceptions

<code>CMSEException</code>
----------------------------

Implements **cms::Session** (p. 3315).

6.69.2.21 `virtual cms::TemporaryTopic* ac-  
tivismq::core::ActiveMQSession::createTemporaryTopic ( )  
throw ( cms::CMSEException ) [virtual]`

Creates a TemporaryTopic object.

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3316).

6.69.2.22 `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage  
( ) throw ( cms::CMSEException ) [virtual]`

Creates a new TextMessage.

### Returns

a newly created TextMessage.

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3316).

6.69.2.23 `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage  
( const std::string & text ) throw ( cms::CMSEException ) [virtual]`

Creates a new TextMessage and set the text to the value given.

### Parameters

<i>text</i>	- The initial text for the message
-------------	------------------------------------

### Returns

a newly created TextMessage with the given Text set in the Message body.

### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3316).

6.69.2.24 `virtual cms::Topic* activemq::core::ActiveMQSession::createTopic ( const  
std::string & topicName ) throw ( cms::CMSEException ) [virtual]`

Creates a topic identity given a Queue name.

### Parameters

<i>topicName</i>	- the name of the new Topic
------------------	-----------------------------

**Exceptions**

<i>CMSException</i>
---------------------

Implements **cms::Session** (p. 3317).

6.69.2.25 `void activemq::core::ActiveMQSession::deliverAcks ( )`

Request that this Session inform all of its consumers to deliver their pending acks.

6.69.2.26 `virtual void activemq::core::ActiveMQSession::dispatch ( const Pointer< MessageDispatch > & message ) [virtual]`

Dispatches a message to a particular consumer.

**Parameters**

<i>message</i>	- the message to be dispatched
----------------	--------------------------------

Implements **activemq::core::Dispatcher** (p. 1750).

6.69.2.27 `void activemq::core::ActiveMQSession::doStartTransaction ( ) throw ( exceptions::ActiveMQException )`

Starts if not already start a Transaction for this Session.

If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

**Exceptions**

<i>ActiveMQException</i>	if this is not a Transacted Session.
--------------------------	--------------------------------------

6.69.2.28 `void activemq::core::ActiveMQSession::fire ( const exceptions::ActiveMQException & ex )`

Fires the given exception to the exception listener of the connection.

6.69.2.29 `virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode ( ) const throw ( cms::CMSException ) [virtual]`

Returns the acknowledgment mode of the session.

**Returns**

the Sessions Acknowledge Mode

Implements **cms::Session** (p. 3317).

**6.69.2.30** **ActiveMQConnection\*** **activemq::core::ActiveMQSession::getConnection** ( )  
**const** [inline]

Gets the **ActiveMQConnection** (p. 244) that is associated with this session.

**6.69.2.31** **cms::ExceptionListener\*** **activemq::core::ActiveMQSession::getExceptionListener**  
( )

This method gets any registered exception listener of this sessions connection and returns it.

Mainly intended for use by the objects that this session creates so that they can notify the client of exceptions that occur in the context of another thread.

#### Returns

**cms::ExceptionListener** (p. 1801) pointer or NULL

**6.69.2.32** **long long** **activemq::core::ActiveMQSession::getLastDeliveredSequenceId** ( ) **const**  
[inline]

Gets the currently set Last Delivered Sequence Id.

#### Returns

long long containing the sequence id of the last delivered Message.

**6.69.2.33** **Pointer<commands::ConsumerId>** **activemq::core::ActiveMQSession::getNextConsumerId** ( )

Get the Next available Consumer Id.

#### Returns

the next id in the sequence.

**6.69.2.34** **Pointer<commands::ProducerId>** **activemq::core::ActiveMQSession::getNextProducerId** ( )

Get the Next available Producer Id.

#### Returns

the next id in the sequence.

**6.69.2.35** `const commands::SessionId& activemq::core::ActiveMQSession::getSessionId ( ) const [inline]`

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

#### Returns

SessionId Reference

**6.69.2.36** `const commands::SessionInfo& activemq::core::ActiveMQSession::getSessionInfo ( ) const [inline]`

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

#### Returns

SessionInfo Reference

**6.69.2.37** `Pointer<ActiveMQTransactionContext> activemq::core::ActiveMQSession::getTransactionContext ( ) [inline]`

Gets the Pointer to this Session's TransactionContext.

#### Returns

a Pointer to this Session's TransactionContext

**6.69.2.38** `bool activemq::core::ActiveMQSession::isAutoAcknowledge ( ) const [inline]`

References cms::Session::AUTO\_ACKNOWLEDGE.

**6.69.2.39** `bool activemq::core::ActiveMQSession::isClientAcknowledge ( ) const [inline]`

References cms::Session::CLIENT\_ACKNOWLEDGE.

**6.69.2.40** `bool activemq::core::ActiveMQSession::isDupsOkAcknowledge ( ) const [inline]`

References cms::Session::DUPS\_OK\_ACKNOWLEDGE.



6.69.2.41 `bool activemq::core::ActiveMQSession::isIndividualAcknowledge ( ) const`  
`[inline]`

References `cms::Session::INDIVIDUAL_ACKNOWLEDGE`.

6.69.2.42 `bool activemq::core::ActiveMQSession::isStarted ( ) const`

Indicates whether or not the session is currently in the started state.

6.69.2.43 `virtual bool activemq::core::ActiveMQSession::isTransacted ( ) const throw (`  
`cms::CMSException ) [virtual]`

Gets if the Sessions is a Transacted Session.

#### Returns

transacted true - false.

Implements `cms::Session` (p. 3317).

6.69.2.44 `void activemq::core::ActiveMQSession::oneway ( Pointer<`  
`commands::Command > command ) throw (`  
`activemq::exceptions::ActiveMQException )`

Sends a oneway message.

#### Parameters

<i>command</i>	The message to send.
----------------	----------------------

#### Exceptions

<i>ActiveMQException</i>	if not currently connected, or if the operation fails for any reason.
--------------------------	---

6.69.2.45 `virtual void activemq::core::ActiveMQSession::recover ( ) throw (`  
`cms::CMSException ) [virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"

- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

### Exceptions

<i>CMSException</i>	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<i>IllegalStateException</i>	- if the method is called by a transacted session.

Implements **cms::Session** (p. 3318).

6.69.2.46 `void activemq::core::ActiveMQSession::redispatch ( MessageDispatchChannel & unconsumedMessages )`

Redispatches the given set of unconsumed messages to the consumers.

### Parameters

<i>unconsumedMessages</i>	- unconsumed messages to be redelivered.
---------------------------	--

6.69.2.47 `void activemq::core::ActiveMQSession::removeConsumer ( const Pointer< commands::ConsumerId > & consumerId, long long lastDeliveredSequenceId = 0 ) throw ( activemq::exceptions::ActiveMQException )`

Dispose of a MessageConsumer from this session.

Removes it from the Connection and clean up any resources associated with it.

### Parameters

<i>consumerId</i>	The ConsumerId of the MessageConsumer to remove from this Session.
<i>lastDeliveredSequenceId</i>	The sequenceId of the last Message the consumer delivered.

### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.69.2.48 `void activemq::core::ActiveMQSession::removeProducer ( const Pointer< commands::ProducerId > & producerId ) throw ( activemq::exceptions::ActiveMQException )`

Dispose of a MessageProducer from this session.

Removes it from the Connection and clean up any resources associated with it.

#### Parameters

<i>producerId</i>	The ProducerId of the MessageProducer to remove from this session.
-------------------	--

#### Exceptions

<i>ActiveMQException</i>	if an internal error occurs.
--------------------------	------------------------------

6.69.2.49 `virtual void activemq::core::ActiveMQSession::rollback ( ) throw ( cms::CMSEException ) [virtual]`

Rollsback all messages done in this transaction and releases any locks currently held.

#### Exceptions

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3318).

6.69.2.50 `void activemq::core::ActiveMQSession::send ( cms::Message * message, ActiveMQProducer * producer, util::Usage * usage ) throw ( cms::CMSEException )`

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.

Asynchronous sends will be chosen if at all possible.

#### Parameters

<i>message</i>	The message to send to the broker.
<i>producer</i>	The sending Producer
<i>usage</i>	Pointer to a Usage tracker which if set will be increased by the size of the given message.

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.69.2.51 `void activemq::core::ActiveMQSession::setLastDeliveredSequenceId ( long long value ) [inline]`

Sets the value of the Last Delivered Sequence Id.

**Parameters**

<i>value</i>	The new value to assign to the Last Delivered Sequence Id property.
--------------	---

6.69.2.52 `void activemq::core::ActiveMQSession::start ( )`

Stops asynchronous message delivery.

6.69.2.53 `void activemq::core::ActiveMQSession::stop ( )`

Starts asynchronous message delivery.

6.69.2.54 `void activemq::core::ActiveMQSession::syncRequest ( Pointer<  
commands::Command > command, unsigned int timeout = 0 ) throw (  
activemq::exceptions::ActiveMQException )`

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

**Parameters**

<i>command</i>	The request command.
<i>timeout</i>	The time to wait for a response, default is zero or infinite.

**Exceptions**

<i>ActiveMQException</i>	thrown if an error response was received from the broker, or if any other error occurred.
--------------------------	---

6.69.2.55 `virtual void activemq::core::ActiveMQSession::unsubscribe ( const std::string & name  
) throw ( cms::CMSEException )` `[virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

**Parameters**

<i>name</i>	the name used to identify this subscription
-------------	---

**Exceptions**

<i>CMSEException</i>	
----------------------	--

Implements **cms::Session** (p. 3319).

6.69.2.56 void `activemq::core::ActiveMQSession::wakeup ( )`

Causes the Session to wakeup its executor and ensure all messages are dispatched.

### 6.69.3 Friends And Related Function Documentation

6.69.3.1 friend class `ActiveMQSessionExecutor` [`friend`]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSession.h`

## 6.70 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

```
#include <src/main/activemq/core/ActiveMQSessionExecutor.h>
```

Inheritance diagram for `activemq::core::ActiveMQSessionExecutor`:

### Public Member Functions

- **ActiveMQSessionExecutor** (**ActiveMQSession** \*session)  
*Creates an un-started executor for the given session.*
- virtual ~**ActiveMQSessionExecutor** ()  
*Calls **stop()** (p. 506) then **clear()** (p. 504).*
- virtual void **execute** (const **Pointer**< **MessageDispatch** > &data)  
*Executes the dispatch.*
- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)  
*Executes the dispatch.*
- virtual void **clearMessagesInProgress** ()  
*Removes all messages in the Dispatch Channel so that non are delivered.*
- virtual bool **hasUnconsumedMessages** () const
- virtual void **wakeup** ()  
*wakeup this executor and dispatch any pending messages.*
- virtual void **start** ()  
*Starts the dispatching.*
- virtual void **stop** ()  
*Stops dispatching.*
- virtual void **close** ()  
*Terminates the dispatching thread.*
- virtual bool **isRunning** () const

- virtual bool **isEmpty** ()
- virtual void **clear** ()  
*Removes all queued messages and destroys them.*
- virtual bool **iterate** ()  
*Iterates on the **MessageDispatchChannel** (p. 2559) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer**< **MessageDispatch** > > **getUnconsumedMessages** ()

### 6.70.1 Detailed Description

Delegate dispatcher for a single session.

Contains a thread to provide for asynchronous dispatching.

### 6.70.2 Constructor & Destructor Documentation

#### 6.70.2.1 `activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor ( ActiveMQSession * session )`

Creates an un-started executor for the given session.

#### 6.70.2.2 `virtual activemq::core::ActiveMQSessionExecutor::~~ActiveMQSessionExecutor ( )` [virtual]

Calls **stop()** (p. 506) then **clear()** (p. 504).

### 6.70.3 Member Function Documentation

#### 6.70.3.1 `virtual void activemq::core::ActiveMQSessionExecutor::clear ( )` [inline, virtual]

Removes all queued messages and destroys them.

#### 6.70.3.2 `virtual void activemq::core::ActiveMQSessionExecutor::clearMessagesInProgress ( )` [inline, virtual]

Removes all messages in the Dispatch Channel so that non are delivered.

#### 6.70.3.3 `virtual void activemq::core::ActiveMQSessionExecutor::close ( )` [inline, virtual]

Terminates the dispatching thread.

Once this is called, the executor is no longer usable.

6.70.3.4 `virtual void activemq::core::ActiveMQSessionExecutor::execute ( const Pointer< MessageDispatch > & data ) [virtual]`

Executes the dispatch.

Adds the given data to the end of the queue.

#### Parameters

<i>data</i>	- the data to be dispatched.
-------------	------------------------------

6.70.3.5 `virtual void activemq::core::ActiveMQSessionExecutor::executeFirst ( const Pointer< MessageDispatch > & data ) [virtual]`

Executes the dispatch.

Adds the given data to the beginning of the queue.

#### Parameters

<i>data</i>	- the data to be dispatched.
-------------	------------------------------

6.70.3.6 `std::vector< Pointer< MessageDispatch > > activemq::core::ActiveMQSessionExecutor::getUnconsumedMessages ( ) [inline]`

#### Returns

a vector containing all the unconsumed messages, this clears the Message Dispatch Channel when called.

6.70.3.7 `virtual bool activemq::core::ActiveMQSessionExecutor::hasUnconsumedMessages ( ) const [inline, virtual]`

#### Returns

true if there are any pending messages in the dispatch channel.

6.70.3.8 `virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ( ) [inline, virtual]`

#### Returns

true if there are no messages in the Dispatch Channel.

6.70.3.9 `virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ( ) const`  
`[inline, virtual]`

#### Returns

true indicates if the executor is started

6.70.3.10 `virtual bool activemq::core::ActiveMQSessionExecutor::iterate ( ) [virtual]`

Iterates on the **MessageDispatchChannel** (p. 2559) sending all pending messages to the Consumers they are destined for.

#### Returns

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 3679).

6.70.3.11 `virtual void activemq::core::ActiveMQSessionExecutor::start ( ) [virtual]`

Starts the dispatching.

6.70.3.12 `virtual void activemq::core::ActiveMQSessionExecutor::stop ( ) [virtual]`

Stops dispatching.

6.70.3.13 `virtual void activemq::core::ActiveMQSessionExecutor::wakeup ( ) [virtual]`

wakeup this executor and dispatch any pending messages.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSessionExecutor.h`

## 6.71 **activemq::commands::ActiveMQStreamMessage** Class Reference

```
#include <src/main/activemq/commands/ActiveMQStreamMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQStreamMessage`:



## Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual **~ActiveMQStreamMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQStreamMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()  
*Store the Data that was written to the stream before a send.*
- virtual **cms::StreamMessage** \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **reset** () throw ( cms::CMSException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*
- virtual bool **readBoolean** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a Boolean from the Stream message stream.*
- virtual void **writeBoolean** (bool value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a boolean to the Stream message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a Byte from the Stream message stream.*
- virtual void **writeByte** (unsigned char value) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a byte to the Stream message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a byte array from the Stream message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw ( cms::MessageNotWriteableException, cms::CMSException )

*Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.*

- virtual int **readBytes** (unsigned char \*buffer, int length) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a portion of the Stream message stream.*

- virtual void **writeBytes** (const unsigned char \*value, int offset, int length) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a portion of a byte array to the Stream message stream.*

- virtual char **readChar** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a Char from the Stream message stream.*

- virtual void **writeChar** (char value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a char to the Stream message stream as a 1-byte value.*

- virtual float **readFloat** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit float from the Stream message stream.*

- virtual void **writeFloat** (float value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a float to the Stream message stream as a 4 byte value.*

- virtual double **readDouble** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit double from the Stream message stream.*

- virtual void **writeDouble** (double value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a double to the Stream message stream as a 8 byte value.*

- virtual short **readShort** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit signed short from the Stream message stream.*

- virtual void **writeShort** (short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed short to the Stream message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit unsigned short from the Stream message stream.*

- virtual void **writeUnsignedShort** (unsigned short value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a unsigned short to the Stream message stream as a 2 byte value.*

- virtual int **readInt** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit signed integer from the Stream message stream.*

- virtual void **writeInt** (int value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed int to the Stream message stream as a 4 byte value.*

- virtual long long **readLong** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit long from the Stream message stream.*

- virtual void **writeLong** (long long value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a long long to the Stream message stream as a 8 byte value.*

- virtual std::string **readString** () const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )

*Reads an ASCII String from the Stream message stream.*

- virtual void **writeString** (const std::string &value) throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes an ASCII String to the Stream message stream.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQSTREAMMESSAGE** = 27

## 6.71.1 Constructor & Destructor Documentation

6.71.1.1 **activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage ( )**

6.71.1.2 **virtual activemq::commands::ActiveMQStreamMessage::~~ActiveMQStreamMessage ( )** [virtual]

## 6.71.2 Member Function Documentation

6.71.2.1 **virtual void activemq::commands::ActiveMQStreamMessage::clearBody ( )** throw ( cms::CMSEException ) [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 398).

6.71.2.2 **virtual cms::StreamMessage\* activemq::commands::ActiveMQStreamMessage::clone ( )** const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

## Returns

new copy of this message

Implements **cms::Message** (p. 2498).

6.71.2.3 `virtual ActiveMQStreamMessage* activemq::commands::ActiveMQStreamMessage::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2480).

6.71.2.4 `virtual void activemq::commands::ActiveMQStreamMessage::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2481).

6.71.2.5 `virtual bool activemq::commands::ActiveMQStreamMessage::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 399).

6.71.2.6 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Message** (p. 2483).

6.71.2.7 `virtual void activemq::commands::ActiveMQStreamMessage::onSend ( )`  
`[virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 407).

6.71.2.8 `virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean ( ) const`  
`throw ( cms::MessageEOFException, cms::MessageFormatException,`  
`cms::MessageNotReadableException, cms::CMSEException )`  
`[virtual]`

Reads a Boolean from the Stream message stream.

#### Returns

boolean value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements `cms::StreamMessage` (p. 3597).

6.71.2.9 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte`  
`( ) const throw ( cms::MessageEOFException,`  
`cms::MessageFormatException, cms::MessageNotReadableException,`  
`cms::CMSEException ) [virtual]`

Reads a Byte from the Stream message stream.

#### Returns

unsigned char value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.

<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3598).

6.71.2.10 `virtual int activemq::commands::ActiveMQStreamMessage::readBytes ( unsigned char * buffer, int length ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an CMSException is thrown. No bytes will be read from the stream for this exception case.

#### Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3599).

6.71.2.11 `virtual int activemq::commands::ActiveMQStreamMessage::readBytes ( std::vector< unsigned char > & value ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

#### Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3598).

6.71.2.12 `virtual char activemq::commands::ActiveMQStreamMessage::readChar ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a Char from the Stream message stream.

#### Returns

char value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
---------------------	---

<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3600).

```
6.71.2.13 virtual double activemq::commands::ActiveMQStreamMessage::readDouble ( ) const
throw ( cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException )
[virtual]
```

Reads a 64 bit double from the Stream message stream.

#### Returns

double value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3601).

```
6.71.2.14 virtual float activemq::commands::ActiveMQStreamMessage::readFloat ( ) const
throw ( cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException )
[virtual]
```

Reads a 32 bit float from the Stream message stream.

#### Returns

double value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
---------------------	---



<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3601).

6.71.2.15 `virtual int activemq::commands::ActiveMQStreamMessage::readInt ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 32 bit signed integer from the Stream message stream.

#### Returns

int value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3602).

6.71.2.16 `virtual long long activemq::commands::ActiveMQStreamMessage::readLong ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [virtual]`

Reads a 64 bit long from the Stream message stream.

#### Returns

long long value from stream

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to read the message due to some internal error.
---------------------	---

<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3602).

```
6.71.2.17 virtual short activemq::commands::ActiveMQStreamMessage::readShort ( ) const
throw ( cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSEException )
[virtual]
```

Reads a 16 bit signed short from the Stream message stream.

#### Returns

short value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3603).

```
6.71.2.18 virtual std::string activemq::commands::ActiveMQStreamMessage::readString
( ) const throw ( cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSEException ) [virtual]
```

Reads an ASCII String from the Stream message stream.

#### Returns

String from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
----------------------	---

<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3603).

6.71.2.19 virtual unsigned short activemq::commands::ActiveMQStreamMessage::readUnsignedShort ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException ) [virtual]

Reads a 16 bit unsigned short from the Stream message stream.

#### Returns

unsigned short value from stream

#### Exceptions

<i>CMSEException</i>	- if the CMS provider fails to read the message due to some internal error.
<i>MessageEOFException</i>	- if unexpected end of message stream has been reached.
<i>MessageFormatException</i>	- if this type conversion is invalid.
<i>MessageNotReadableException</i>	- if the message is in write-only mode.

Implements **cms::StreamMessage** (p. 3604).

6.71.2.20 virtual void activemq::commands::ActiveMQStreamMessage::reset ( ) throw ( cms::CMSEException ) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

#### Exceptions

<i>CMSEException</i>	
----------------------	--

6.71.2.21 `virtual std::string activemq::commands::ActiveMQStreamMessage::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2490).

6.71.2.22 `virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean ( bool`  
`value ) throw ( cms::MessageNotWriteableException, cms::CMSException`  
`) [virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

#### Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3605).

6.71.2.23 `virtual void activemq::commands::ActiveMQStreamMessage::writeByte (`  
`unsigned char value ) throw ( cms::MessageNotWriteableException,`  
`cms::CMSException ) [virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

#### Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3605).

```
6.71.2.24 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes
( const unsigned char * value, int offset, int length ) throw (
  cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3605).

```
6.71.2.25 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes
( const std::vector< unsigned char > & value ) throw (
  cms::MessageNotWriteableException, cms::CMSException )
[virtual]
```

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3606).

6.71.2.26 `virtual void activemq::commands::ActiveMQStreamMessage::writeChar ( char value )  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[virtual]`

Writes a char to the Stream message stream as a 1-byte value.

#### Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3606).

6.71.2.27 `virtual void activemq::commands::ActiveMQStreamMessage::writeDouble ( double  
value ) throw ( cms::MessageNotWriteableException, cms::CMSException  
 ) [virtual]`

Writes a double to the Stream message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3607).

6.71.2.28 `virtual void activemq::commands::ActiveMQStreamMessage::writeFloat ( float value )  
throw ( cms::MessageNotWriteableException, cms::CMSException )  
[virtual]`

Writes a float to the Stream message stream as a 4 byte value.

#### Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3607).

```
6.71.2.29 virtual void activemq::commands::ActiveMQStreamMessage::writeInt ( int value )
          throw ( cms::MessageNotWriteableException, cms::CMSException )
          [virtual]
```

Writes a signed int to the Stream message stream as a 4 byte value.

**Parameters**

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3608).

```
6.71.2.30 virtual void activemq::commands::ActiveMQStreamMessage::writeLong ( long long
          value ) throw ( cms::MessageNotWriteableException, cms::CMSException
          ) [virtual]
```

Writes a long long to the Stream message stream as a 8 byte value.

**Parameters**

<i>value</i>	signed long long to write to the stream
--------------	---

**Exceptions**

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3608).

6.71.2.31 `virtual void activemq::commands::ActiveMQStreamMessage::writeShort ( short value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a signed short to the Stream message stream as a 2 byte value.

#### Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3608).

6.71.2.32 `virtual void activemq::commands::ActiveMQStreamMessage::writeString ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes an ASCII String to the Stream message stream.

#### Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3609).

6.71.2.33 `virtual void activemq::commands::ActiveMQStreamMessage::writeUnsignedShort ( unsigned short value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

#### Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------



## 6.72 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller

### Class Reference 523

#### Exceptions

<i>CMSException</i>	- if the CMS provider fails to write the message due to some internal error.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode.

Implements **cms::StreamMessage** (p. 3609).

### 6.71.3 Field Documentation

6.71.3.1 `const unsigned char activemq::commands::ActiveMQStreamMessage::ID_ - ACTIVEMQSTREAMMESSAGE = 27` [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQStreamMessage.h**

## 6.72 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 523).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.72.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 523).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.72.2 Constructor & Destructor Documentation

6.72.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
 ( ) [inline]

6.72.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
 ( ) [inline, virtual]

### 6.72.3 Member Function Documentation

6.72.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.72.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

## 6.72 ac-

**activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

**Class Reference**

**525**

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.72.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
[virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2658).

**6.72.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

6.72.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

6.72.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2660).

## 6.73 ac-

ativemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller

### Class Reference

527

```
6.72.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2661).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQStreamMessageMarshaller.h**

## 6.73 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 527).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMar
```

Inheritance diagram for **ativemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs)  
throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.73.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 527).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.73.2 Constructor & Destructor Documentation

6.73.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]

6.73.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.73.3 Member Function Documentation

6.73.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

### 6.73 ac-

**activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

#### Class Reference

529

6.73.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.73.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2671).

6.73.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

6.73.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

6.73.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673).



## 6.74 ac-

ativemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller

### Class Reference

531

```
6.73.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightUnmarshal  
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (  
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2674).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQStreamMessageMarshaller.h**

## 6.74 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 531).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMar
```

Inheritance diagram for **ativemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs)  
throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.74.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 531).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.74.2 Constructor & Destructor Documentation

6.74.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]

6.74.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.74.3 Member Function Documentation

6.74.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.74 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**

### Class Reference

533

6.74.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.74.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2667).

6.74.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

**6.74.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

**6.74.3.6** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

## 6.75 ac-

ativemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

### Class Reference

535

```
6.74.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQStreamMessageMarshaller.h**

## 6.75 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 535).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMar
```

Inheritance diagram for **ativemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs)  
throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.75.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 535).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.75.2 Constructor & Destructor Documentation

6.75.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]

6.75.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.75.3 Member Function Documentation

6.75.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.75 ac-

ativemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

### Class Reference

537

6.75.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.75.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2654).

6.75.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.75.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

6.75.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).



## 6.76 ac-

ativemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller

### Class Reference

539

```
6.75.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQStreamMessageMarshaller.h**

## 6.76 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 539).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMar
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.76.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 539).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.76.2 Constructor & Destructor Documentation

6.76.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]

6.76.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.76.3 Member Function Documentation

6.76.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.76 ac-

**ativemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

### Class Reference

541

6.76.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.76.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

6.76.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

6.76.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

6.76.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

## 6.77 ac-

ativemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller

### Class Reference

543

```
6.76.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2665).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQStreamMessageMarshaller.h**

## 6.77 activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 543).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMar
```

Inheritance diagram for **ativemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**:

### Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs)  
throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.77.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 543).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.77.2 Constructor & Destructor Documentation

6.77.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller**  
( ) [inline]

6.77.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller**  
( ) [inline, virtual]

### 6.77.3 Member Function Documentation

6.77.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.77 ac-

**activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**

### Class Reference

545

6.77.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.77.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

6.77.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

6.77.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

6.77.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).



6.77.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2678).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQStreamMessageMarshaller.h**

## 6.78 activemq::commands::ActiveMQTempDestination Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempDestination.h>
```

Inheritance diagram for activemq::commands::ActiveMQTempDestination:

#### Public Member Functions

- **ActiveMQTempDestination** ()
- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** ()
- virtual unsigned char **getDataStructureType** () const  
Get the **DataStructure** (p. 1628) Type as defined in *CommandTypes.h*.
- virtual **ActiveMQTempDestination** \* **cloneDataStructure** () const  
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSet** \*value) const  
*Compares the **DataSet** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual void **close** () throw ( cms::CMSEException )  
*Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.*
- void **setConnection** (core::ActiveMQConnection \*connection)  
*Sets the Parent Connection that is notified when this destination is destroyed.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEMPDESTINATION** = 0

### Protected Attributes

- core::ActiveMQConnection \* **connection**  
*Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.*

## 6.78.1 Constructor & Destructor Documentation

- 6.78.1.1 `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ( )`
- 6.78.1.2 `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ( const std::string & name )`
- 6.78.1.3 `virtual activemq::commands::ActiveMQTempDestination::~~ActiveMQTempDestination ( ) [virtual]`

## 6.78.2 Member Function Documentation

- 6.78.2.1 `virtual ActiveMQTempDestination* activemq::commands::ActiveMQTempDestination::cloneDataSet ( ) const [inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 296).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 575), and **activemq::commands::ActiveMQTempTopic** (p. 604).

**6.78.2.2** `virtual void activemq::commands::ActiveMQTempDestination::close ( ) throw ( cms::CMSException ) [virtual]`

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.

This should only be called when the user is certain that they are finished with this destination. The TempDestination is not closed automatically on shutdown. throws **cms::CMSException** (p. 1130)

Implements **cms::Closeable** (p. 1120).

**6.78.2.3** `virtual void activemq::commands::ActiveMQTempDestination::copyDataStructure ( const DataStructure * src ) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 296).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 576), and **activemq::commands::ActiveMQTempTopic** (p. 604).

References **activemq::commands::ActiveMQDestination::copyDataStructure()**.

**6.78.2.4** `virtual bool activemq::commands::ActiveMQTempDestination::equals ( const DataStructure * value ) const [inline, virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 297).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 576), and **activemq::commands::ActiveMQTempTopic** (p. 605).

References **activemq::commands::ActiveMQDestination::equals()**.

6.78.2.5 `virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType ( ) const [virtual]`

Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 298).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 577), and **activemq::commands::ActiveMQTempTopic** (p. 605).

6.78.2.6 `void activemq::commands::ActiveMQTempDestination::setConnection ( core::ActiveMQConnection * connection ) [inline]`

Sets the Parent Connection that is notified when this destination is destroyed.

#### Parameters

<i>connection</i>	- The parent connection.
-------------------	--------------------------

6.78.2.7 `virtual std::string activemq::commands::ActiveMQTempDestination::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 302).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 578), and **activemq::commands::ActiveMQTempTopic** (p. 606).

### 6.78.3 Field Documentation

6.78.3.1 `core::ActiveMQConnection* activemq::commands::ActiveMQTempDestination::connection [protected]`

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.79 **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**  
**Class Reference** 551  
6.78.3.2 `const unsigned char activemq::commands::ActiveMQTempDestination::ID_-`  
`ACTIVEMQTEMPDESTINATION = 0 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempDestination.h`

## 6.79 **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** **Class Reference**

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 551).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller`:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.79.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 551).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.79.2 Constructor & Destructor Documentation

6.79.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.79.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

### 6.79.3 Member Function Documentation

6.79.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 305).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 580), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 608).

6.79.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## 6.79 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller

### Class Reference 553

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 306).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 580), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 609).

```
6.79.3.3 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 306).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 581), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 609).

```
6.79.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 307).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 581), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 610).

```
6.79.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 307).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 582), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 610).



## 6.80 ac-

**activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

### Class Reference

555

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempDestinationMarshaller.h**

## 6.80 **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 555).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.80.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 555).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.80.2 Constructor & Destructor Documentation

6.80.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller ( ) [inline]`

6.80.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller ( ) [inline, virtual]`

## 6.80.3 Member Function Documentation

6.80.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 309).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 584), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 616).

6.80.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**6.80 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**  
**Class Reference** **557**  
**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 310).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 584), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 617).

**6.80.3.3** `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 310).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 585), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 617).

**6.80.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 311).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 585), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 618).

```
6.80.3.5 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 311).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 586), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 618).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempDestinationMarshaller.h**

## 6.81 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 558).

## 6.81 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

### Class Reference

559

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`:

## Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.81.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 558).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

### 6.81.2 Constructor & Destructor Documentation

6.81.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller** ( ) [`inline`]

6.81.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller** ( ) [`inline`, `virtual`]

### 6.81.3 Member Function Documentation

6.81.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
`[virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 313).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 588), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 612).

6.81.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
`[virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 314).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 588), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 613).

## 6.81 ac-

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

### Class Reference

561

```
6.81.3.3 virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMarshal1  
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 314).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 589), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 613).

```
6.81.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMarshal2  
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 315).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 589), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 614).

**6.81.3.5** `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 315).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 590), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 614).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h`

## 6.82 **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 562).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**:

#### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()



## 6.82 ac-

**activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

### Class Reference

563

- virtual **~ActiveMQTempDestinationMarshaller** ( )
- virtual void **tightUnmarshal** ( **OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** ( **OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** ( **OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** ( **OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** ( **OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.82.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 562).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.82.2 Constructor & Destructor Documentation

6.82.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller** ( ) [inline]

6.82.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller** ( ) [inline, virtual]

### 6.82.3 Member Function Documentation

6.82.3.1 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::looseMarshal** ( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 317).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 592), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 620).

```
6.82.3.2 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 318).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 592), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 621).

```
6.82.3.3 virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

## 6.82 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller

### Class Reference 565

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 318).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 593), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 621).

```
6.82.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 319).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 593), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 622).

6.82.3.5 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 319).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 594), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempDestinationMarshaller.h**

## 6.83 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 566).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller:

#### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (OpenWireFormat \**wireFormat*, commands::DataStructure \**dataStructure*, decaf::io::DataInputStream \**dataIn*, utils::BooleanStream \**bs*) throw ( decaf::io::IOException )

### 6.83 ac-

## activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller

### Class Reference

567

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.83.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 566).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.83.2 Constructor & Destructor Documentation

6.83.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller**  
( ) [inline]

6.83.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller**  
( ) [inline, virtual]

### 6.83.3 Member Function Documentation

6.83.3.1 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure,  
**decaf::io::DataOutputStream** \* dataOut ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 321).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 596), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 624).

**6.83.3.2** `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 322).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 596), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 625).

**6.83.3.3** `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.83 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller

### Class Reference

#### Returns

569

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 322).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 597), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 625).

6.83.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 323).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 597), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 626).

6.83.3.5 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 323).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 598), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 626).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempDestinationMarshaller.h**

## 6.84 **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 570).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDes
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**:

### Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



## 6.84 ac-

### activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller

#### Class Reference

571

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.84.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 570).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.84.2 Constructor & Destructor Documentation

6.84.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller**  
( ) [inline]

6.84.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::~~ActiveMQTempDestinationMarshaller**  
( ) [inline, virtual]

#### 6.84.3 Member Function Documentation

6.84.3.1 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( decaf::io::IOException )  
[virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 325).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 600), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 628).

```
6.84.3.2 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 326).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 600), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 629).

```
6.84.3.3 virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.84 ac-

**activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**

### Class Reference

573

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 326).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 601), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 629).

6.84.3.4 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightMarshal2 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut, utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 327).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 601), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 630).

6.84.3.5 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 327).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 602), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 630).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h`

## 6.85 `activemq::commands::ActiveMQTempQueue` Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempQueue.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempQueue`:

### Public Member Functions

- `ActiveMQTempQueue ()`
- `ActiveMQTempQueue (const std::string &name)`
- `virtual ~ActiveMQTempQueue ()`
- `virtual unsigned char getDataStructureType () const`  
*Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.*
- `virtual ActiveMQTempQueue * cloneDataStructure () const`  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- `virtual void copyDataStructure (const DataStructure *src)`  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- `virtual std::string toString () const`  
*Converts the Destination Name into a String.*
- `virtual bool equals (const DataStructure *value) const`  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- `virtual const cms::Destination * getCMSDestination () const`
- `virtual cms::Destination::DestinationType getDestinationType () const`  
*Retrieve the Destination Type for this Destination.*
- `virtual cms::Destination * clone () const`  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- `virtual void copy (const cms::Destination &source)`  
*Copies the contents of the given Destination object to this one.*
- `virtual const cms::CMSProperties & getCMSProperties () const`

*Retrieve any properties that might be part of the destination that was specified.*

- virtual std::string **getQueueName** () const throw ( cms::CMSEException )

*Gets the name of this queue.*

- virtual void **destroy** () throw ( cms::CMSEException )

*Destroy's the Temp Destination at the Broker.*

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEMPQUEUE** = 102

## 6.85.1 Constructor & Destructor Documentation

6.85.1.1 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ( )`

6.85.1.2 `activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ( const std::string & name )`

6.85.1.3 `virtual activemq::commands::ActiveMQTempQueue::~~ActiveMQTempQueue ( )`  
[virtual]

## 6.85.2 Member Function Documentation

6.85.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTempQueue::clone ( )`  
`const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

### Returns

cloned copy of this object

Implements **cms::Destination** (p. 1690).

6.85.2.2 `virtual ActiveMQTempQueue* activemq::commands::ActiveMQTempQueue::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 548).

6.85.2.3 `virtual void activemq::commands::ActiveMQTempQueue::copy ( const cms::Destination & source ) [inline, virtual]`

Copies the contents of the given Destination object to this one.

#### Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1690).

6.85.2.4 `virtual void activemq::commands::ActiveMQTempQueue::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 549).

6.85.2.5 `virtual void activemq::commands::ActiveMQTempQueue::destroy ( ) throw ( cms::CMSEException ) [virtual]`

Destroy's the Temp Destination at the Broker.

#### Exceptions

<i>CMSEException</i>
----------------------

Implements **cms::TemporaryQueue** (p. 3702).

6.85.2.6 `virtual bool activemq::commands::ActiveMQTempQueue::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 549).

```
6.85.2.7 virtual const cms::Destination* ac-
        tivemq::commands::ActiveMQTempQueue::getCMSDestination (
        ) const [inline, virtual]
```

**Returns**

the **cms::Destination** (p. 1688) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 298).

```
6.85.2.8 virtual const cms::CMSProperties& ac-
        tivemq::commands::ActiveMQTempQueue::getCMSProperties ( )
        const [inline, virtual]
```

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

**Returns**

const reference to a properties object.

Implements **cms::Destination** (p. 1690).

```
6.85.2.9 virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType
        ( ) const [virtual]
```

Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.

**Returns**

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 550).

```
6.85.2.10 virtual cms::Destination::DestinationType
        activemq::commands::ActiveMQTempQueue::getDestinationType ( ) const
        [inline, virtual]
```

Retrieve the Destination Type for this Destination.

**Returns**

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 298).

References cms::Destination::TEMPORARY\_QUEUE.

6.85.2.11 `virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName ( )  
const throw ( cms::CMSException ) [inline, virtual]`

Gets the name of this queue.

#### Returns

The queue name.

Implements **cms::TemporaryQueue** (p. 3703).

6.85.2.12 `virtual std::string activemq::commands::ActiveMQTempQueue::toString ( ) const  
[virtual]`

Converts the Destination Name into a String.

#### Returns

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 550).

### 6.85.3 Field Documentation

6.85.3.1 `const unsigned char activemq::commands::ActiveMQTempQueue::ID_-  
ACTIVEMQTEMPQUEUE = 102 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempQueue.h`

## 6.86 **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 578).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`:

### Public Member Functions

- **ActiveMQTempQueueMarshaller ( )**



6.86

activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller

Class Reference

579

- virtual `~ActiveMQTempQueueMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.86.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 578).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.86.2 Constructor & Destructor Documentation

6.86.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller ( ) [inline]`

6.86.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller ( ) [inline, virtual]`

### 6.86.3 Member Function Documentation

6.86.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.86.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.86.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 552).

```
6.86.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.86

**activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller****Class Reference****581**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 552).

6.86.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 553).

6.86.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 554).

```
6.86.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 554).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempQueueMarshaller.h**

## 6.87 **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 582).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**:

**Public Member Functions**

- **ActiveMQTempQueueMarshaller** ()

6.87

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller

Class Reference

583

- virtual `~ActiveMQTempQueueMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.87.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 582).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.87.2 Constructor & Destructor Documentation

6.87.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller ( ) [inline]`

6.87.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller ( ) [inline, virtual]`

### 6.87.3 Member Function Documentation

6.87.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.87.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.87.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556).

```
6.87.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.87

**activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller****Class Reference****585**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556).

6.87.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 557).

6.87.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 557).

```
6.87.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 558).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempQueueMarshaller.h**

## 6.88 **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 586).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**:

**Public Member Functions**

- **ActiveMQTempQueueMarshaller** ()



6.88

activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller

Class Reference

587

- virtual `~ActiveMQTempQueueMarshaller ()`
- virtual `commands::DataStructure * createObject ()` const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType ()` const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)` throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)` throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)` throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)` throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)` throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.88.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 586).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.88.2 Constructor & Destructor Documentation

6.88.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller ( )` [inline]

6.88.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller ( )` [inline, virtual]

### 6.88.3 Member Function Documentation

6.88.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::createObject ( )` const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.88.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.88.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 560).

```
6.88.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.88

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller****Class Reference****589**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 560).

6.88.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 561).

6.88.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 561).

```
6.88.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 562).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempQueueMarshaller.h**

## 6.89 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 590).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**:

**Public Member Functions**

- **ActiveMQTempQueueMarshaller** ()

6.89

activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller

Class Reference

591

- virtual `~ActiveMQTempQueueMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.89.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 590).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.89.2 Constructor & Destructor Documentation

6.89.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller ( ) [inline]`

6.89.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller ( ) [inline, virtual]`

### 6.89.3 Member Function Documentation

6.89.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.89.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.89.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 563).

```
6.89.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.89

**activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**

**Class Reference**

**593**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 564).

6.89.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 564).

6.89.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 565).

```
6.89.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 566).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempQueueMarshaller.h**

## 6.90 **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 594).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**:

**Public Member Functions**

- **ActiveMQTempQueueMarshaller** ()



## 6.90

activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller

### Class Reference

595

- virtual `~ActiveMQTempQueueMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.90.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 594).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.90.2 Constructor & Destructor Documentation

6.90.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller ( ) [inline]`

6.90.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller ( ) [inline, virtual]`

### 6.90.3 Member Function Documentation

6.90.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.90.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.90.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 567).

```
6.90.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.90

**activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

**Class Reference**

**597**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 568).

6.90.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 568).

6.90.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 569).

```
6.90.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 569).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempQueueMarshaller.h**

## 6.91 **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 598).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**:

**Public Member Functions**

- **ActiveMQTempQueueMarshaller** ()

## 6.91

activemq:wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller

### Class Reference

599

- virtual `~ActiveMQTempQueueMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.91.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 598).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.91.2 Constructor & Destructor Documentation

6.91.2.1 `activemq:wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller ( ) [inline]`

6.91.2.2 `virtual activemq:wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller ( ) [inline, virtual]`

### 6.91.3 Member Function Documentation

6.91.3.1 `virtual commands::DataStructure* activemq:wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.91.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::getDataStructureType( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaller.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.91.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseMarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 571).

```
6.91.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::looseUnmarshal( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

6.91

**activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller****Class Reference****601**

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 572).

6.91.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 572).

6.91.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 573).

```
6.91.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 573).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempQueueMarshaller.h**

**6.92 activemq::commands::ActiveMQTempTopic Class Reference**

```
#include <src/main/activemq/commands/ActiveMQTempTopic.h>
```

Inheritance diagram for **activemq::commands::ActiveMQTempTopic**:

**Public Member Functions**

- **ActiveMQTempTopic** ()
- **ActiveMQTempTopic** (const std::string &name)
- virtual ~**ActiveMQTempTopic** ()
- virtual unsigned char **getDataStructureType** () const

Get the **DataStructure** (p. 1628) Type as defined in *CommandTypes.h*.



- virtual **ActiveMQTempTopic** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Converts the Destination Name into a String.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** \* **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual **cms::Destination** \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)  
*Copies the contents of the given Destination object to this one.*
- virtual const **cms::CMSProperties** & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getTopicName** () const throw ( cms::CMSException )  
*Gets the name of this topic.*
- virtual void **destroy** () throw ( cms::CMSException )  
*Destroy's the Temp Destination at the Broker.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEMPTOPIC** = 103

## 6.92.1 Constructor & Destructor Documentation

6.92.1.1 **activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic** ( )

6.92.1.2 **activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic** ( const std::string &  
*name* )

6.92.1.3 **virtual activemq::commands::ActiveMQTempTopic::~~ActiveMQTempTopic** ( )  
[virtual]

## 6.92.2 Member Function Documentation

6.92.2.1 **virtual cms::Destination\*** **activemq::commands::ActiveMQTempTopic::clone** ( )  
const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

**Returns**

cloned copy of this object

Implements **cms::Destination** (p. 1690).

```
6.92.2.2 virtual ActiveMQTempTopic* ac-
          tivemq::commands::ActiveMQTempTopic::cloneDataStructure ( )
          const [virtual]
```

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 548).

```
6.92.2.3 virtual void activemq::commands::ActiveMQTempTopic::copy ( const
          cms::Destination & source ) [inline, virtual]
```

Copies the contents of the given Destination object to this one.

**Parameters**

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1690).

```
6.92.2.4 virtual void activemq::commands::ActiveMQTempTopic::copyDataStructure ( const
          DataStructure * src ) [virtual]
```

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns**

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 549).

```
6.92.2.5 virtual void activemq::commands::ActiveMQTempTopic::destroy ( ) throw (
          cms::CMSEException ) [virtual]
```

Destroy's the Temp Destination at the Broker.

### Exceptions

<i>CMSException</i>	
---------------------	--

Implements **cms::TemporaryTopic** (p. 3704).

**6.92.2.6** `virtual bool activemq::commands::ActiveMQTempTopic::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 549).

**6.92.2.7** `virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination ( ) const [inline, virtual]`

### Returns

the **cms::Destination** (p. 1688) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 298).

**6.92.2.8** `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempTopic::getCMSProperties ( ) const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

### Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1690).

**6.92.2.9** `virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType ( ) const [virtual]`

Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.

**Returns**

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 550).

```
6.92.2.10 virtual cms::Destination::DestinationType
activemq::commands::ActiveMQTempTopic::getDestinationType ( ) const
[inline, virtual]
```

Retrieve the Destination Type for this Destination.

**Returns**

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 298).

References cms::Destination::TEMPORARY\_TOPIC.

```
6.92.2.11 virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName ( )
const throw ( cms::CMSException ) [inline, virtual]
```

Gets the name of this topic.

**Returns**

The topic name.

Implements **cms::TemporaryTopic** (p. 3704).

```
6.92.2.12 virtual std::string activemq::commands::ActiveMQTempTopic::toString ( ) const
[virtual]
```

Converts the Destination Name into a String.

**Returns**

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 550).

**6.92.3 Field Documentation**

```
6.92.3.1 const unsigned char activemq::commands::ActiveMQTempTopic::ID_ -
ACTIVEMQTEMPTOPIC = 103 [static]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTempTopic.h**

6.93

activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller

Class Reference

6.93 ~~activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller~~ 607

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 607).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.93.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 607).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.93.2 Constructor & Destructor Documentation

6.93.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]`

6.93.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ( ) [inline, virtual]`

### 6.93.3 Member Function Documentation

6.93.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.93.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.93.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.93

**activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller****Class Reference****609****Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 552).

**6.93.3.4** `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 552).

**6.93.3.5** `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 553).

```
6.93.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 554).

```
6.93.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 554).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempTopicMarshaller.h**



6.94

activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller

Class Reference

6.94 — activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller <sup>611</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 611).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.94.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 611).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.94.2 Constructor & Destructor Documentation

6.94.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]`

6.94.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ( ) [inline, virtual]`

## 6.94.3 Member Function Documentation

6.94.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.94.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.94.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.94

**activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

**Class Reference**

613

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 560).

6.94.3.4 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 560).

6.94.3.5 **virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal1 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 561).

**6.94.3.6** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 561).

**6.94.3.7** virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 562).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempTopicMarshaller.h**

6.95

activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller

Class Reference

6.95 — activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller <sup>615</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 615).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.95.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 615).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.95.2 Constructor & Destructor Documentation

6.95.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]`

6.95.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ( ) [inline, virtual]`

## 6.95.3 Member Function Documentation

6.95.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.95.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.95.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.95

**activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

**Class Reference**

617

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556).

6.95.3.4 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )** [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556).

6.95.3.5 **virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal1 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, utils::BooleanStream \* bs ) throw ( decaf::io::IOException )** [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 557).

6.95.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 557).

6.95.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 558).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempTopicMarshaller.h**



6.96

activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller

Class Reference

6.96 — activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller

619

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 619).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.96.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 619).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.96.2 Constructor & Destructor Documentation

6.96.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]`

6.96.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ( ) [inline, virtual]`

## 6.96.3 Member Function Documentation

6.96.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.96.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.96.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.96

**activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**

**Class Reference**

**621**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 563).

6.96.3.4 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
[virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 564).

6.96.3.5 **virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal1**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 564).

**6.96.3.6** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 565).

**6.96.3.7** virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 566).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempTopicMarshaller.h**

6.97

activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller

Class Reference

6.97 ~~activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller~~ <sup>623</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 623).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller ()**
- virtual **~ActiveMQTempTopicMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.97.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 623).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.97.2 Constructor & Destructor Documentation

6.97.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]`

6.97.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ( ) [inline, virtual]`

## 6.97.3 Member Function Documentation

6.97.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.97.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.97.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.97

**activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

**Class Reference**

**625**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 567).

6.97.3.4 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
[virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 568).

6.97.3.5 **virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal1**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 568).

6.97.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 569).

6.97.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 569).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempTopicMarshaller.h**



6.98

activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller

Class Reference

6.98 ~~activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller~~ <sup>627</sup>

## Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 627).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller:

### Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.98.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 627).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.98.2 Constructor & Destructor Documentation

6.98.2.1 `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller ( ) [inline]`

6.98.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller ( ) [inline, virtual]`

## 6.98.3 Member Function Documentation

6.98.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.98.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.98.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

6.98

**activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**

**Class Reference**

629

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 571).

6.98.3.4 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
[virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 572).

6.98.3.5 **virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal1**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 572).

**6.98.3.6** virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 573).

**6.98.3.7** virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 573).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTempTopicMarshaller.h**

## 6.99 activemq::commands::ActiveMQTextMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQTextMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQTextMessage:

### Public Member Functions

- **ActiveMQTextMessage** ()
- virtual **~ActiveMQTextMessage** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ActiveMQTextMessage** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw ( cms::CMSException )  
*Clears out the body of the message.*
- virtual void **beforeMarshal** (wireformat::WireFormat \*wireFormat) throw ( decaf::io::IOException )  
*Performs any cleanup or other tasks that must be done before the **Message** (p. 2475) is marshalled to its binary WireFormat version.*
- virtual unsigned int **getSize** () const  
*Returns the Size of this message in Bytes.*
- virtual cms::TextMessage \* **clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual std::string **getText** () const throw ( cms::CMSException )  
*Gets the message character buffer.*
- virtual void **setText** (const char \*msg) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets the message contents, does not take ownership of the passed char\*, but copies it instead.*
- virtual void **setText** (const std::string &msg) throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Sets the message contents.*

## Data Fields

- `std::auto_ptr< std::string > text`

## Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTEXTMESSAGE** = 28

## 6.99.1 Constructor & Destructor Documentation

6.99.1.1 `activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage ( )`

6.99.1.2 `virtual activemq::commands::ActiveMQTextMessage::~~ActiveMQTextMessage ( )`  
[virtual]

## 6.99.2 Member Function Documentation

6.99.2.1 `virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal ( wireformat::WireFormat * wireFormat ) throw ( decaf::io::IOException )`  
[virtual]

Performs any cleanup or other tasks that must be done before the **Message** (p. 2475) is marshalled to its binary WireFormat version.

### Parameters

<i>wireFormat</i>	the WireFormat instance that is marshalling this message.
-------------------	---

Implements **activemq::wireformat::MarshalAware** (p. 2445).

6.99.2.2 `virtual void activemq::commands::ActiveMQTextMessage::clearBody ( ) throw ( cms::CMSEException )` [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 398).

6.99.2.3 `virtual cms::TextMessage* activemq::commands::ActiveMQTextMessage::clone ( ) const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

### Returns

new copy of this message

Implements **cms::Message** (p. 2498).

```
6.99.2.4 virtual ActiveMQTextMessage* ac-
        tivemq::commands::ActiveMQTextMessage::cloneDataStructure (
        ) const [virtual]
```

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 2480).

```
6.99.2.5 virtual void activemq::commands::ActiveMQTextMessage::copyDataStructure ( const
        DataStructure * src ) [virtual]
```

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 2481).

```
6.99.2.6 virtual bool activemq::commands::ActiveMQTextMessage::equals ( const
        DataStructure * value ) const [virtual]
```

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 399).

```
6.99.2.7 virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType
        ( ) const [virtual]
```

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Message** (p. 2483).

```
6.99.2.8 virtual unsigned int activemq::commands::ActiveMQTextMessage::getSize ( ) const
        [virtual]
```

Returns the Size of this message in Bytes.

**Returns**

number of bytes this message equates to.

Reimplemented from **activemq::commands::Message** (p. 2485).

```
6.99.2.9 virtual std::string activemq::commands::ActiveMQTextMessage::getText ( ) const
        throw ( cms::CMSException ) [virtual]
```

Gets the message character buffer.

**Returns**

The message character buffer.

**Exceptions**

<i>CMSException</i>	- if an internal error occurs.
---------------------	--------------------------------

Implements **cms::TextMessage** (p. 3705).

```
6.99.2.10 virtual void activemq::commands::ActiveMQTextMessage::setText ( const std::string &
        msg ) throw ( cms::MessageNotWriteableException, cms::CMSException
        ) [virtual]
```

Sets the message contents.

**Parameters**

<i>msg</i>	The message buffer.
------------	---------------------

**Exceptions**

<i>CMSException</i>	- if an internal error occurs.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 3706).



## 6.100

### activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference 635

6.99.2.11 `virtual void activemq::commands::ActiveMQTextMessage::setText ( const char * msg ) throw ( cms::MessageNotWriteableException, cms::CMSException )`  
[virtual]

Sets the message contents, does not take ownership of the passed char\*, but copies it instead.

#### Parameters

<i>msg</i>	The message buffer.
------------	---------------------

#### Exceptions

<i>CMSException</i>	- if an internal error occurs.
<i>MessageNotWriteableException</i>	- if the message is in read-only mode..

Implements **cms::TextMessage** (p. 3706).

6.99.2.12 `virtual std::string activemq::commands::ActiveMQTextMessage::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2490).

### 6.99.3 Field Documentation

6.99.3.1 `const unsigned char activemq::commands::ActiveMQTextMessage::ID_ - ACTIVEMQTEXTMESSAGE = 28` [static]

6.99.3.2 `std::auto_ptr<std::string> activemq::commands::ActiveMQTextMessage::text`  
[mutable]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTextMessage.h**

## 6.100 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 635).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMes
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`:

## Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.100.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 635).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

### 6.100.2 Constructor & Destructor Documentation

- 6.100.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller`  
( ) [inline]

## 6.100

**activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

### Class Reference

637

6.100.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::~ActiveMQTextMessageMarshaller ( ) [inline, virtual]`

## 6.100.3 Member Function Documentation

6.100.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.100.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.100.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2658).

```
6.100.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

```
6.100.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659).

## 6.100

**activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

### Class Reference

639

```
6.100.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2660).

```
6.100.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2661).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTextMessageMarshaller.h**

## 6.101 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 640).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller:

### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.101.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 640).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.101

**activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

**Class Reference**

**641**

### 6.101.2 Constructor & Destructor Documentation

6.101.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
( ) [inline]

6.101.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
( ) [inline, virtual]

### 6.101.3 Member Function Documentation

6.101.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.101.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.101.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2667).

6.101.3.4 **virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

6.101.3.5 **virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )**  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.101

**activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

### Class Reference

643

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2668).

```
6.101.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

```
6.101.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTextMessageMarshaller.h**

## 6.102 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 644).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller:

### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.102.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 644).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.102

**activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

**Class Reference**

**645**

### 6.102.2 Constructor & Destructor Documentation

6.102.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
( ) [inline]

6.102.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
( ) [inline, virtual]

### 6.102.3 Member Function Documentation

6.102.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.102.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.102.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2671).

```
6.102.3.4  virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

```
6.102.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.102

**activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

### Class Reference

647

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672).

```
6.102.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673).

```
6.102.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2674).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTextMessageMarshaller.h**

## 6.103 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 648).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller:

### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.103.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 648).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.103

**activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

### Class Reference

649

#### 6.103.2 Constructor & Destructor Documentation

6.103.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
( ) [inline]

6.103.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
( ) [inline, virtual]

#### 6.103.3 Member Function Documentation

6.103.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.103.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.103.3.3 **virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2654).

```
6.103.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

```
6.103.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.103

**activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

### Class Reference

651

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2655).

```
6.103.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

```
6.103.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTextMessageMarshaller.h**

## 6.104 activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 652).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller:

### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.104.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 652).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.104

**activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**

### Class Reference

653

#### 6.104.2 Constructor & Destructor Documentation

6.104.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
( ) [inline]

6.104.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
( ) [inline, virtual]

#### 6.104.3 Member Function Documentation

6.104.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.104.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.104.3.3 **virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

```
6.104.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676).

```
6.104.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.104

**activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller**

### Class Reference

655

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

6.104.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677).

6.104.3.7 `virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2678).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTextMessageMarshaller.h**

## 6.105 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 656).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMes
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller:

### Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.105.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 656).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.105

**activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

**Class Reference**

**657**

### 6.105.2 Constructor & Destructor Documentation

6.105.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller**  
( ) [inline]

6.105.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller**  
( ) [inline, virtual]

### 6.105.3 Member Function Documentation

6.105.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.105.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.105.3.3 **virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

```
6.105.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2663).

```
6.105.3.5  virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.105

**activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

### Class Reference

659

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.105.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664).

```
6.105.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2665).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTextMessageMarshaller.h**

## 6.106 activemq::commands::ActiveMQTopic Class Reference

```
#include <src/main/activemq/commands/ActiveMQTopic.h>
```

Inheritance diagram for activemq::commands::ActiveMQTopic:

### Public Member Functions

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual ~**ActiveMQTopic** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.*
- virtual **ActiveMQTopic** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination \* **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const  
*Retrieve the Destination Type for this Destination.*
- virtual cms::Destination \* **clone** () const  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const cms::Destination &source)  
*Copies the contents of the given Destination object to this one.*
- virtual const cms::CMSProperties & **getCMSProperties** () const  
*Retrieve any properties that might be part of the destination that was specified.*
- virtual std::string **getTopicName** () const throw ( cms::CMSException )  
*Gets the name of this topic.*

### Static Public Attributes

- static const unsigned char **ID\_ACTIVEMQTOPIC** = 101

## 6.106.1 Constructor & Destructor Documentation

6.106.1.1 `activemq::commands::ActiveMQTopic::ActiveMQTopic ( )`

6.106.1.2 `activemq::commands::ActiveMQTopic::ActiveMQTopic ( const std::string & name )`

6.106.1.3 `virtual activemq::commands::ActiveMQTopic::~~ActiveMQTopic ( ) [virtual]`

## 6.106.2 Member Function Documentation

6.106.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTopic::clone ( ) const [inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

### Returns

cloned copy of this object

Implements **cms::Destination** (p. 1690).

6.106.2.2 `virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 296).

6.106.2.3 `virtual void activemq::commands::ActiveMQTopic::copy ( const cms::Destination & source ) [inline, virtual]`

Copies the contents of the given Destination object to this one.

### Parameters

<i>source</i>	The source Destination object.
---------------	--------------------------------

Implements **cms::Destination** (p. 1690).

6.106.2.4 `virtual void activemq::commands::ActiveMQTopic::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 296).

6.106.2.5 `virtual bool activemq::commands::ActiveMQTopic::equals ( const DataStructure * value ) const [inline, virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 297).

References `activemq::commands::ActiveMQDestination::equals()`.

6.106.2.6 `virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination ( ) const [inline, virtual]`

#### Returns

the **cms::Destination** (p. 1688) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 298).

6.106.2.7 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties ( ) const [inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

#### Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1690).

6.106.2.8 `virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType ( ) const [virtual]`

Get the **DataSet** (p. 1628) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 298).

6.106.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType ( ) const [inline, virtual]`

Retrieve the Destination Type for this Destination.

#### Returns

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 298).

References cms::Destination::TOPIC.

6.106.2.10 `virtual std::string activemq::commands::ActiveMQTopic::getTopicName ( ) const throw ( cms::CMSExcption ) [inline, virtual]`

Gets the name of this topic.

#### Returns

The topic name.

Implements **cms::Topic** (p. 3758).

6.106.2.11 `virtual std::string activemq::commands::ActiveMQTopic::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 302).

### 6.106.3 Field Documentation

6.106.3.1 `const unsigned char activemq::commands::ActiveMQTopic::ID_ -  
ACTIVEMQTOPIC = 101` `[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTopic.h`

## 6.107 `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 664).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMa
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual `~ActiveMQTopicMarshaller` ()
- virtual `commands::DataStructure * createObject` () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType` () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`) throw ( `decaf::io::IOException` )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.107.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 664).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.107.2 Constructor & Destructor Documentation

6.107.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
( ) [inline]

6.107.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
( ) [inline, virtual]

### 6.107.3 Member Function Documentation

6.107.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.107.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.107.3.3  virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 305).

```
6.107.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 306).

```
6.107.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.



#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 306).

```
6.107.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 307).

```
6.107.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 307).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTopicMarshaller.h**

## 6.108 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 668).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.108.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 668).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.108.2 Constructor & Destructor Documentation

6.108.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
( ) [inline]

6.108.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
( ) [inline, virtual]

### 6.108.3 Member Function Documentation

6.108.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.108.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```

6.108.3.3  virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 313).

```

6.108.3.4  virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 314).

```

6.108.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 314).

```
6.108.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 315).

```
6.108.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 315).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTopicMarshaller.h**

## 6.109 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 672).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.109.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 672).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.109.2 Constructor & Destructor Documentation

6.109.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
( ) [inline]

6.109.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
( ) [inline, virtual]

### 6.109.3 Member Function Documentation

6.109.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.109.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.109.3.3  virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 309).

```
6.109.3.4  virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 310).

```
6.109.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.



## 6.109 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller

### Class Reference 675

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 310).

```
6.109.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 311).

```
6.109.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 311).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTopicMarshaller.h**

## 6.110 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 676).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.110.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 676).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.110.2 Constructor & Destructor Documentation

6.110.2.1 **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
( ) [inline]

6.110.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
( ) [inline, virtual]

### 6.110.3 Member Function Documentation

6.110.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.110.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.110.3.3  virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 317).

```
6.110.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 318).

```
6.110.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 318).

```
6.110.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 319).

```
6.110.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 319).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTopicMarshaller.h**

## 6.111 activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 680).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller**:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.111.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 680).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.111.2 Constructor & Destructor Documentation

6.111.2.1 **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
( ) [inline]

6.111.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
( ) [inline, virtual]

### 6.111.3 Member Function Documentation

6.111.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.111.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.111.3.3  virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 325).

```
6.111.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 326).

```
6.111.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.



## 6.111 activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller

### Class Reference 683

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 326).

```
6.111.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 327).

```
6.111.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 327).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ActiveMQTopicMarshaller.h**

## 6.112 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 684).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**:

### Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.112.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 684).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.112.2 Constructor & Destructor Documentation

6.112.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller**  
 ( ) [inline]

6.112.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller**  
 ( ) [inline, virtual]

### 6.112.3 Member Function Documentation

6.112.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.112.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.112.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 321).

```
6.112.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 322).

```
6.112.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 322).

```
6.112.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 323).

```
6.112.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 323).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTopicMarshaller.h**

## 6.113 activemq::core::ActiveMQTransactionContext Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

```
#include <src/main/activemq/core/ActiveMQTransactionContext.h>
```

### Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** \*session, const **decaf::util::Properties** &properties)  
*Constructor.*
- virtual ~**ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer**< **Synchronization** > &sync)  
*Adds a **Synchronization** (p. 3659) to this Transaction.*
- virtual void **removeSynchronization** (const **Pointer**< **Synchronization** > &sync)  
*Removes a **Synchronization** (p. 3659) to this Transaction.*
- virtual void **begin** () throw ( exceptions::ActiveMQException )  
*Begins a new transaction if one is not currently in progress.*
- virtual void **commit** () throw ( exceptions::ActiveMQException )  
*Commit the current Transaction.*
- virtual void **rollback** () throw ( exceptions::ActiveMQException )  
*Rollback the current Transaction.*
- virtual const **decaf::lang::Pointer**< **commands::TransactionId** > & **getTransactionId** () const  
*Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.*
- virtual bool **isInTransaction** () const  
*Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.*

### 6.113.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

#### Since

2.0

### 6.113.2 Constructor & Destructor Documentation

6.113.2.1 `activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext (   
 ActiveMQSession * session, const decaf::util::Properties & properties )`

Constructor.

#### Parameters

<i>session</i>	The session that contains this transaction
<i>properties</i>	Configuration parameters for this object

6.113.2.2 `virtual activemq::core::ActiveMQTransactionContext::~~ActiveMQTransactionContext (   
 ) [virtual]`

### 6.113.3 Member Function Documentation

6.113.3.1 `virtual void activemq::core::ActiveMQTransactionContext::addSynchronization ( const   
 Pointer< Synchronization > & sync ) [virtual]`

Adds a **Synchronization** (p. 3659) to this Transaction.

#### Parameters

<i>sync</i>	- The <b>Synchronization</b> (p. 3659) instance to add.
-------------	---

6.113.3.2 `virtual void activemq::core::ActiveMQTransactionContext::begin ( ) throw (   
 exceptions::ActiveMQException ) [virtual]`

Begins a new transaction if one is not currently in progress.

#### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

6.113.3.3 `virtual void activemq::core::ActiveMQTransactionContext::commit ( ) throw ( exceptions::ActiveMQException ) [virtual]`

Commit the current Transaction.

#### Exceptions

<i>ActiveMQException</i>
--------------------------

6.113.3.4 `virtual const decaf::lang::Pointer<commands::TransactionId>& activemq::core::ActiveMQTransactionContext::getTransactionId ( ) const [virtual]`

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

#### Returns

TransactionInfo

#### Exceptions

<i>InvalidStateException</i> if a Transaction is not in progress.
---

6.113.3.5 `virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction ( ) const [virtual]`

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

#### Returns

true if a transaction is in progress.

6.113.3.6 `virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization ( const Pointer< Synchronization > & sync ) [virtual]`

Removes a **Synchronization** (p. 3659) to this Transaction.

#### Parameters

<i>sync</i>	- The <b>Synchronization</b> (p. 3659) instance to add.
-------------	---



6.113.3.7 virtual void activemq::core::ActiveMQTransactionContext::rollback ( ) throw ( exceptions::ActiveMQException ) [virtual]

Rollback the current Transaction.

### Exceptions

ActiveMQException
-------------------

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQTransactionContext.h

## 6.114 decaf::util::zip::Adler32 Class Reference

Clas that can be used to compute an Adler-32 **Checksum** (p. 1114) for a data stream.

```
#include <src/main/decaf/util/zip/Adler32.h>
```

Inheritance diagram for decaf::util::zip::Adler32:

### Public Member Functions

- **Adler32** ()
- virtual ~**Adler32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()  
*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)  
*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (int byte)  
*Updates the current checksum with the specified byte value.*

### 6.114.1 Detailed Description

Clas that can be used to compute an Adler-32 **Checksum** (p. 1114) for a data stream.

The Alder-32 checksum trades reliability for speed over the CRC-32 algorithm.

**Since**

1.0

**6.114.2 Constructor & Destructor Documentation**6.114.2.1 `decaf::util::zip::Adler32::Adler32 ( )`6.114.2.2 `virtual decaf::util::zip::Adler32::~~Adler32 ( )` [virtual]**6.114.3 Member Function Documentation**6.114.3.1 `virtual long long decaf::util::zip::Adler32::getValue ( ) const` [virtual]**Returns**

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 1115).6.114.3.2 `virtual void decaf::util::zip::Adler32::reset ( )` [virtual]

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 1115).6.114.3.3 `virtual void decaf::util::zip::Adler32::update ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Updates the current checksum with the specified array of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

**Exceptions**

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if <code>offset + length &gt; size</code> of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1115).

6.114.3.4 `virtual void decaf::util::zip::Adler32::update ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Updates the current checksum with the specified array of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if <code>offset + length &gt; size of the buffer</code> .
----------------------------------	---

Implements **decaf::util::zip::Checksum** (p. 1116).

6.114.3.5 `virtual void decaf::util::zip::Adler32::update ( const std::vector< unsigned char > & buffer ) [virtual]`

Updates the current checksum with the specified vector of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implements **decaf::util::zip::Checksum** (p. 1116).

6.114.3.6 `virtual void decaf::util::zip::Adler32::update ( int byte ) [virtual]`

Updates the current checksum with the specified byte value.

#### Parameters

<i>byte</i>	The byte value to update the current <b>Checksum</b> (p. 1114) with (0..255).
-------------	---

Implements **decaf::util::zip::Checksum** (p. 1116).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Adler32.h`

## 6.115 decaf::lang::Appendable Class Reference

An object to which char sequences and values can be appended.

```
#include <src/main/decaf/lang/Appendable.h>
```

Inheritance diagram for decaf::lang::Appendable:

## Public Member Functions

- virtual **~Appendable** ()
- virtual **Appendable & append** (char value)=0 throw ( decaf::lang::Exception )  
*Appends the specified character to this **Appendable** (p. 693).*
- virtual **Appendable & append** (const **CharSequence** \*csq)=0 throw ( decaf::lang::Exception )  
*Appends the specified character sequence to this **Appendable** (p. 693).*
- virtual **Appendable & append** (const **CharSequence** \*csq, int start, int end)=0 throw ( decaf::lang::Exception )  
*Appends a subsequence of the specified character sequence to this **Appendable** (p. 693).*

### 6.115.1 Detailed Description

An object to which char sequences and values can be appended.

The **Appendable** (p. 693) interface must be implemented by any class whose instances are intended to receive formatted output from a Formatter.

TODO The characters to be appended should be valid Unicode characters as described in Unicode **Character** (p. 1069) Representation. Note that supplementary characters may be composed of multiple 16-bit char values.

Appendables are not necessarily safe for multithreaded access. **Thread** (p. 3707) safety is the responsibility of classes that extend and implement this interface.

Since this interface may be implemented by existing classes with different styles of error handling there is no guarantee that errors will be propagated to the invoker.

**Since**

1.0

### 6.115.2 Constructor & Destructor Documentation

6.115.2.1 virtual decaf::lang::Appendable::~~Appendable ( ) [inline, virtual]

### 6.115.3 Member Function Documentation

6.115.3.1 virtual **Appendable&** decaf::lang::Appendable::append ( char value ) throw ( decaf::lang::Exception ) [pure virtual]

Appends the specified character to this **Appendable** (p. 693).

**Parameters**

<i>value</i>	The character to append.
--------------	--------------------------

**Returns**

a Reference to this **Appendable** (p. 693)

**Exceptions**

<b>Exception</b> (p. 1794)	if an error occurs.
----------------------------	---------------------

Implemented in **decaf::io::Writer** (p. 3953), and **decaf::nio::CharBuffer** (p. 1094).

6.115.3.2 virtual **Appendable&** decaf::lang::Appendable::append ( const CharSequence \* *csq*, int *start*, int *end* ) throw ( decaf::lang::Exception ) [pure virtual]

Appends a subsequence of the specified character sequence to this **Appendable** (p. 693).

**Parameters**

<i>csq</i>	- The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
<i>start</i>	The index of the first character in the subsequence.
<i>end</i>	The index of the character following the last character in the subsequence.

**Returns**

a Reference to this **Appendable** (p. 693)

**Exceptions**

<b>Exception</b> (p. 1794)	if an error occurs.
<i>IndexOutOfBoundsException</i>	<i>start</i> is greater than <i>end</i> , or <i>end</i> is greater than <i>csq.length()</i>

Implemented in **decaf::io::Writer** (p. 3953), and **decaf::nio::CharBuffer** (p. 1093).

6.115.3.3 virtual **Appendable&** decaf::lang::Appendable::append ( const CharSequence \* *csq* ) throw ( decaf::lang::Exception ) [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 693).

**Parameters**

<i>csq</i>	The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
------------	---

**Returns**

a Reference to this **Appendable** (p. 693).

**Exceptions**

<b>Exception</b> (p. 1794) if an error occurs.
--

Implemented in **decaf::io::Writer** (p. 3953), and **decaf::nio::CharBuffer** (p. 1094).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Appendable.h**

**6.116 decaf::internal::AprPool Class Reference**

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

```
#include <src/main/decaf/internal/AprPool.h>
```

**Public Member Functions**

- **AprPool** ()
- virtual **~AprPool** ()
- apr\_pool\_t \* **getAprPool** () const  
*Gets the internal APR Pool.*
- void **cleanup** ()  
*Clears data that was allocated by this pool.*

**Static Public Member Functions**

- static apr\_pool\_t \* **getGlobalPool** ()  
*Gets a pointer to the Global APR Pool used for the Application.*

**6.116.1 Detailed Description**

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

**6.116.2 Constructor & Destructor Documentation**

**6.116.2.1 decaf::internal::AprPool::AprPool ( )**

**6.116.2.2 virtual decaf::internal::AprPool::~~AprPool ( )** [virtual]

## 6.117 `decaf::lang::ArrayPointer< T, REFCOUNTER >` Class Template Reference

### 6.116.3 Member Function Documentation

#### 6.116.3.1 `void decaf::internal::AprPool::cleanup ( )`

Clears data that was allocated by this pool.

Users should call this after getting the data from the APR functions and copying it to someplace safe.

#### 6.116.3.2 `apr_pool_t* decaf::internal::AprPool::getAprPool ( ) const`

Gets the internal APR Pool.

##### Returns

the internal APR pool

#### 6.116.3.3 `static apr_pool_t* decaf::internal::AprPool::getGlobalPool ( ) [static]`

Gets a pointer to the Global APR Pool used for the Application.

##### Returns

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/AprPool.h`

## 6.117 `decaf::lang::ArrayPointer< T, REFCOUNTER >` Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

### Data Structures

- struct **ArrayData**

### Public Types

- `typedef T * PointerType`
- `typedef T & ReferenceType`
- `typedef const T & ConstReferenceType`
- `typedef REFCOUNTER CounterType`

## Public Member Functions

- **ArrayPointer** ()  
*Default Constructor.*
- **ArrayPointer** (int size)  
*Create a new **ArrayPointer** (p. 697) instance and allocates an internal array that is sized using the passed in size value.*
- **ArrayPointer** (const **PointerType** value, int size)  
*Explicit Constructor, creates an **ArrayPointer** (p. 697) that contains value with a single reference.*
- **ArrayPointer** (const **ArrayPointer** &value) throw ()  
*Copy constructor.*
- virtual ~**ArrayPointer** () throw ()
- void **reset** (T \*value, int size=0)  
*Resets the **ArrayPointer** (p. 697) to hold the new value.*
- T \* **release** ()  
*Releases the **Pointer** (p. 2896) held and resets the internal pointer value to Null.*
- **PointerType** **get** () const  
*Gets the real array pointer that is contained within this **Pointer** (p. 2896).*
- int **length** () const  
*Returns the current size of the contained array or zero if the array is NULL.*
- void **swap** (**ArrayPointer** &value) throw ()  
***Exception** (p. 1794) Safe Swap Function.*
- **ArrayPointer** **clone** () const  
*Creates a new **ArrayPointer** (p. 697) instance that is a clone of the value contained in this **ArrayPointer** (p. 697).*
- **ArrayPointer** & **operator=** (const **ArrayPointer** &right) throw ()  
*Assigns the value of right to this **Pointer** (p. 2896) and increments the reference Count.*
- template<typename T1 , typename R1 >  
**ArrayPointer** & **operator=** (const **ArrayPointer**< T1, R1 > &right) throw ()
- **ReferenceType** **operator[]** (int index)  
*Dereference Operator, returns a reference to the Contained value.*
- **ConstReferenceType** **operator[]** (int index) const
- bool **operator!** () const
- template<typename T1 , typename R1 >  
bool **operator==** (const **ArrayPointer**< T1, R1 > &right) const
- template<typename T1 , typename R1 >  
bool **operator!=** (const **ArrayPointer**< T1, R1 > &right) const

## Friends

- bool **operator==** (const **ArrayPointer** &left, const T \*right)
- bool **operator==** (const T \*left, const **ArrayPointer** &right)
- bool **operator!=** (const **ArrayPointer** &left, const T \*right)
- bool **operator!=** (const T \*left, const **ArrayPointer** &right)



## 6.117 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference

### 6.117.1 Detailed Description

```
template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> class
decaf::lang::ArrayPointer< T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.

This **Pointer** (p. 2896) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of `ReferenceCounter`.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2896) instances in the same manner as normal pointer, except that it does not provide an overload of operators ( `<`, `<=`, `>`, `>=` ). To allow use of a **Pointer** (p. 2896) in a STL container that requires it, **Pointer** (p. 2896) provides an implementation of `std::less`.

Since

1.0

### 6.117.2 Member Typedef Documentation

6.117.2.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef const T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ConstReferenceType`

6.117.2.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER decaf::lang::ArrayPointer< T, REFCOUNTER >::CounterType`

6.117.2.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T* decaf::lang::ArrayPointer< T, REFCOUNTER >::PointerType`

6.117.2.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> typedef T& decaf::lang::ArrayPointer< T, REFCOUNTER >::ReferenceType`

### 6.117.3 Constructor & Destructor Documentation

6.117.3.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayPointer ( ) [inline]`

Default Constructor.

Initialized the contained array pointer to NULL, using the subscript operator results in an exception unless reset to contain a real value.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, and `decaf::lang::ArrayPointer< unsigned char >::reset()`.

```
6.117.3.2  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer<
            T, REFCOUNTER >::ArrayPointer ( int size ) [inline]
```

Create a new **ArrayPointer** (p. 697) instance and allocates an internal array that is sized using the passed in size value.

#### Parameters

<i>size</i>	The size of the array to allocate for this <b>ArrayPointer</b> (p. 697) instance.
-------------	---

```
6.117.3.3  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer<
            T, REFCOUNTER >::ArrayPointer ( const PointerType value, int size )
            [inline, explicit]
```

Explicit Constructor, creates an **ArrayPointer** (p. 697) that contains value with a single reference.

This object now has ownership until a call to release.

#### Parameters

<i>value</i>	The pointer to the instance of the array we are taking ownership of.
<i>size</i>	The size of the array this object is taking ownership of.

```
6.117.3.4  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::ArrayPointer<
            T, REFCOUNTER >::ArrayPointer ( const ArrayPointer< T, REFCOUNTER > &
            value ) throw () [inline]
```

Copy constructor.

Copies the value contained in the **ArrayPointer** (p. 697) to the new instance and increments the reference counter.

```
6.117.3.5  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> virtual
            decaf::lang::ArrayPointer< T, REFCOUNTER >::~~ArrayPointer ( ) throw ()
            [inline, virtual]
```

## 6.117.4 Member Function Documentation

## 6.117 `decaf::lang::ArrayPointer< T, REFCOUNTER >` Class Template Reference 701

6.117.4.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> ArrayPointer  
decaf::lang::ArrayPointer< T, REFCOUNTER >::clone ( ) const [inline]`

Creates a new **ArrayPointer** (p. 697) instance that is a clone of the value contained in this **ArrayPointer** (p. 697).

### Returns

an **ArrayPointer** (p. 697) that contains a copy of the data in this **ArrayPointer** (p. 697).

6.117.4.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> PointerType  
decaf::lang::ArrayPointer< T, REFCOUNTER >::get ( ) const [inline]`

Gets the real array pointer that is contained within this **Pointer** (p. 2896).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2896). Use at your own risk.

### Returns

the contained pointer.

Referenced by `decaf::lang::ArrayPointer< unsigned char >::clone()`, `decaf::lang::ArrayPointerComparator< T, R >::compare()`, `decaf::lang::operator!=()`, `decaf::lang::ArrayPointer< unsigned char >::operator!=()`, `std::less< decaf::lang::ArrayPointer< T > >::operator()()`, `decaf::lang::ArrayPointerComparator< T, R >::operator()()`, `decaf::lang::operator==()`, and `decaf::lang::ArrayPointer< unsigned char >::operator==()`.

6.117.4.3 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> int  
decaf::lang::ArrayPointer< T, REFCOUNTER >::length ( ) const  
[inline]`

Returns the current size of the contained array or zero if the array is NULL.

### Returns

the size of the array or zero if the array is NULL

6.117.4.4 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool  
decaf::lang::ArrayPointer< T, REFCOUNTER >::operator! ( ) const  
[inline]`

```
6.117.4.5  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 ,
           typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator!=
           ( const ArrayPointer< T1, R1 > & right ) const    [inline]
```

```
6.117.4.6  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 ,
           typename R1 > ArrayPointer& decaf::lang::ArrayPointer< T, REFCOUNTER
           >::operator= ( const ArrayPointer< T1, R1 > & right ) throw ()    [inline]
```

```
6.117.4.7  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> ArrayPointer&
           decaf::lang::ArrayPointer< T, REFCOUNTER >::operator= ( const
           ArrayPointer< T, REFCOUNTER > & right ) throw ()    [inline]
```

Assigns the value of **right** to this **Pointer** (p. 2896) and increments the reference Count.

#### Parameters

<i>right</i>	- <b>Pointer</b> (p. 2896) on the right hand side of an operator= call to this.
--------------	---

```
6.117.4.8  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 ,
           typename R1 > bool decaf::lang::ArrayPointer< T, REFCOUNTER >::operator==
           ( const ArrayPointer< T1, R1 > & right ) const    [inline]
```

```
6.117.4.9  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> ReferenceType
           decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] ( int index )
           [inline]
```

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

#### Returns

reference to the contained pointer.

#### Exceptions

<code>NullPointerException</code>	if the contained value is Null
-----------------------------------	--------------------------------

```
6.117.4.10 template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> ConstReferenceType
           decaf::lang::ArrayPointer< T, REFCOUNTER >::operator[] ( int index ) const
           [inline]
```

## 6.117 decaf::lang::ArrayPointer< T, REFCOUNTER > Class Template Reference 703

```
6.117.4.11  template<typename T, typename REFCOUNTER =
             decaf::util::concurrent::atomic::AtomicRefCount> T*
             decaf::lang::ArrayPointer< T, REFCOUNTER >::release ( ) [inline]
```

Releases the **Pointer** (p. 2896) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 2896) is held by more than one object or this method is called from more than one thread.

### Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

### Returns

The pointer instance that was held by this **Pointer** (p. 2896) object, the pointer is no longer owned by this **Pointer** (p. 2896) and won't be freed when this **Pointer** (p. 2896) goes out of scope.

Referenced by decaf::lang::ArrayPointer< unsigned char >::ArrayPointer().

```
6.117.4.12  template<typename T, typename REFCOUNTER =
             decaf::util::concurrent::atomic::AtomicRefCount> void
             decaf::lang::ArrayPointer< T, REFCOUNTER >::reset ( T * value, int size = 0
             ) [inline]
```

Resets the **ArrayPointer** (p. 697) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2896) to a NULL pointer.

### Parameters

<i>value</i>	The new array pointer value to contain.
<i>size</i>	The size of the new array value this object now contains.

```
6.117.4.13  template<typename T, typename REFCOUNTER =
             decaf::util::concurrent::atomic::AtomicRefCount> void
             decaf::lang::ArrayPointer< T, REFCOUNTER >::swap ( ArrayPointer< T,
             REFCOUNTER > & value ) throw () [inline]
```

**Exception** (p. 1794) Safe Swap Function.

### Parameters

<i>value</i>	- the value to swap with this.
--------------	--------------------------------

Referenced by decaf::lang::ArrayPointer< unsigned char >::operator=(), and decaf::lang::ArrayPointer< unsigned char >::swap().

### 6.117.5 Friends And Related Function Documentation

- 6.117.5.1 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= ( const ArrayPointer< T, REFCOUNTER > & left, const T * right ) [friend]`
- 6.117.5.2 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator!= ( const T * left, const ArrayPointer< T, REFCOUNTER > & right ) [friend]`
- 6.117.5.3 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== ( const ArrayPointer< T, REFCOUNTER > & left, const T * right ) [friend]`
- 6.117.5.4 `template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCounter> bool operator== ( const T * left, const ArrayPointer< T, REFCOUNTER > & right ) [friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/ArrayPointer.h`

### 6.118 `decaf::lang::ArrayPointerComparator< T, R >` Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 697).

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Inheritance diagram for `decaf::lang::ArrayPointerComparator< T, R >`:

#### Public Member Functions

- virtual bool **operator()** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const
- virtual int **compare** (const **ArrayPointer**< T, R > &left, const **ArrayPointer**< T, R > &right) const

#### 6.118.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter> class decaf::lang::ArrayPointerComparator<
T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 697).

This allows for a basic ordering to be acheived in Decaf containers.

Custom implementations are possible where an array of some type has a logical natural ordering such as array of integers where the sum of all ints in the array is used.

## 6.118.2 Member Function Documentation

```
6.118.2.1 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual int
decaf::lang::ArrayPointerComparator< T, R >::compare ( const
ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right ) const
[inline, virtual]
```

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

```
6.118.2.2 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool
decaf::lang::ArrayPointerComparator< T, R >::operator() ( const
ArrayPointer< T, R > & left, const ArrayPointer< T, R > & right ) const
[inline, virtual]
```

References decaf::lang::ArrayPointer< T, REFCOUNTER >::get().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

## 6.119 decaf::util::concurrent::atomic::AtomicBoolean Class Reference

A boolean value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>
```

### Public Member Functions

- **AtomicBoolean** ()  
*Creates a new **AtomicBoolean** (p. 705) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)

*Creates a new **AtomicBoolean** (p. 705) with the initial value.*

- virtual **~AtomicBoolean** ()
- bool **get** () const

*Gets the current value of this **AtomicBoolean** (p. 705).*

- void **set** (bool newValue)

*Unconditionally sets to the given value.*

- bool **compareAndSet** (bool expect, bool update)

*Atomically sets the value to the given updated value if the current value == the expected value.*

- bool **getAndSet** (bool newValue)

*Atomically sets to the given value and returns the previous value.*

- std::string **toString** () const

*Returns the String representation of the current value.*

### 6.119.1 Detailed Description

A boolean value that may be updated atomically.

An **AtomicBoolean** (p. 705) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

### 6.119.2 Constructor & Destructor Documentation

#### 6.119.2.1 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ( )`

Creates a new **AtomicBoolean** (p. 705) whose initial value is false.

#### 6.119.2.2 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ( bool initialValue )`

Creates a new **AtomicBoolean** (p. 705) with the initial value.

#### Parameters

<i>initialValue</i>	- The initial value of this boolean.
---------------------	--------------------------------------

#### 6.119.2.3 `virtual decaf::util::concurrent::atomic::AtomicBoolean::~~AtomicBoolean ( )` [inline, virtual]

### 6.119.3 Member Function Documentation

#### 6.119.3.1 `bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet ( bool expect, bool update )`

Atomically sets the value to the given updated value if the current value == the expected value.



**Parameters**

<i>expect</i>	- the expected value
<i>update</i>	- the new value

**Returns**

true if successful. False return indicates that the actual value was not equal to the expected value.

6.119.3.2 `bool decaf::util::concurrent::atomic::AtomicBoolean::get ( ) const` `[inline]`

Gets the current value of this **AtomicBoolean** (p. 705).

**Returns**

the currently set value.

6.119.3.3 `bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet ( bool newValue )`

Atomically sets to the given value and returns the previous value.

**Parameters**

<i>newValue</i>	- the new value
-----------------	-----------------

**Returns**

the previous value

6.119.3.4 `void decaf::util::concurrent::atomic::AtomicBoolean::set ( bool newValue )`  
`[inline]`

Unconditionally sets to the given value.

**Parameters**

<i>newValue</i>	- the new value
-----------------	-----------------

6.119.3.5 `std::string decaf::util::concurrent::atomic::AtomicBoolean::toString ( ) const`

Returns the String representation of the current value.

**Returns**

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicBoolean.h`

## 6.120 `decaf::util::concurrent::atomic::AtomicInteger` Class Reference

An int value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h>
```

Inheritance diagram for `decaf::util::concurrent::atomic::AtomicInteger`:

### Public Member Functions

- **AtomicInteger** ()  
*Create a new **AtomicInteger** (p. 708) with an initial value of 0.*
- **AtomicInteger** (int initialValue)  
*Create a new **AtomicInteger** (p. 708) with the given initial value.*
- virtual **~AtomicInteger** ()
- int **get** () const  
*Gets the current value.*
- void **set** (int newValue)  
*Sets to the given value.*
- int **getAndSet** (int newValue)  
*Atomically sets to the given value and returns the old value.*
- bool **compareAndSet** (int expect, int update)  
*Atomically sets the value to the given updated value if the current value == the expected value.*
- int **getAndIncrement** ()  
*Atomically increments by one the current value.*
- int **getAndDecrement** ()  
*Atomically decrements by one the current value.*
- int **getAndAdd** (int delta)  
*Atomically adds the given value to the current value.*
- int **incrementAndGet** ()  
*Atomically increments by one the current value.*
- int **decrementAndGet** ()  
*Atomically decrements by one the current value.*
- int **addAndGet** (int delta)  
*Atomically adds the given value to the current value.*
- std::string **toString** () const  
*Returns the String representation of the current value.*
- int **intValue** () const

*Description copied from class: Number Returns the value of the specified number as an int.*

- long long **longValue** () const

*Description copied from class: Number Returns the value of the specified number as a long.*

- float **floatValue** () const

*Description copied from class: Number Returns the value of the specified number as a float.*

- double **doubleValue** () const

*Description copied from class: Number Returns the value of the specified number as a double.*

### 6.120.1 Detailed Description

An int value that may be updated atomically.

An **AtomicInteger** (p. 708) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes.

### 6.120.2 Constructor & Destructor Documentation

#### 6.120.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ( )

Create a new **AtomicInteger** (p. 708) with an initial value of 0.

#### 6.120.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ( int *initialValue* )

Create a new **AtomicInteger** (p. 708) with the given initial value.

#### Parameters

<i>initialValue</i>	- The initial value of this object.
---------------------	-------------------------------------

#### 6.120.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~~AtomicInteger ( ) [inline, virtual]

### 6.120.3 Member Function Documentation

#### 6.120.3.1 int decaf::util::concurrent::atomic::AtomicInteger::addAndGet ( int *delta* )

Atomically adds the given value to the current value.

**Parameters**

<i>delta</i>	- the value to add.
--------------	---------------------

**Returns**

the updated value.

6.120.3.2 `bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet ( int expect, int update )`

Atomically sets the value to the given updated value if the current value == the expected value.

**Parameters**

<i>expect</i>	- the expected value
<i>update</i>	- the new value

**Returns**

true if successful. False return indicates that the actual value was not equal to the expected value.

6.120.3.3 `int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ( )`

Atomically decrements by one the current value.

**Returns**

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::release()`.

6.120.3.4 `double decaf::util::concurrent::atomic::AtomicInteger::doubleValue ( ) const`  
`[virtual]`

Description copied from class: `Number` Returns the value of the specified number as a double.

This may involve rounding.

**Returns**

the numeric value represented by this object after conversion to type double.

Implements `decaf::lang::Number` (p. 2787).

6.120.3.5 `float decaf::util::concurrent::atomic::AtomicInteger::floatValue ( ) const`  
[virtual]

Description copied from class: `Number` Returns the value of the specified number as a float.

This may involve rounding.

#### Returns

the numeric value represented by this object after conversion to type float.

Implements **decaf::lang::Number** (p. 2787).

6.120.3.6 `int decaf::util::concurrent::atomic::AtomicInteger::get ( ) const` [inline]

Gets the current value.

#### Returns

the current value.

6.120.3.7 `int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd ( int delta )`

Atomically adds the given value to the current value.

#### Parameters

<i>delta</i>	- The value to add.
--------------	---------------------

#### Returns

the previous value.

6.120.3.8 `int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ( )`

Atomically decrements by one the current value.

#### Returns

the previous value.

6.120.3.9 `int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ( )`

Atomically increments by one the current value.

#### Returns

the previous value.

6.120.3.10 `int decaf::util::concurrent::atomic::AtomicInteger::getAndSet ( int newValue )`

Atomically sets to the given value and returns the old value.

#### Parameters

<i>newValue</i>	- the new value.
-----------------	------------------

#### Returns

the previous value.

6.120.3.11 `int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ( )`

Atomically increments by one the current value.

#### Returns

the updated value.

Referenced by `decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter()`.

6.120.3.12 `int decaf::util::concurrent::atomic::AtomicInteger::intValue ( ) const`  
[virtual]

Description copied from class: `Number` Returns the value of the specified number as an int.

This may involve rounding or truncation.

#### Returns

the numeric value represented by this object after conversion to type int.

Implements **`decaf::lang::Number`** (p. 2788).

6.120.3.13 `long long decaf::util::concurrent::atomic::AtomicInteger::longValue ( ) const`  
[virtual]

Description copied from class: `Number` Returns the value of the specified number as a long.

This may involve rounding or truncation.

#### Returns

the numeric value represented by this object after conversion to type long long.

Implements **`decaf::lang::Number`** (p. 2788).

6.120.3.14 `void decaf::util::concurrent::atomic::AtomicInteger::set ( int newValue )`  
`[inline]`

Sets to the given value.

#### Parameters

<i>newValue</i>	- the new value
-----------------	-----------------

6.120.3.15 `std::string decaf::util::concurrent::atomic::AtomicInteger::toString ( ) const`

Returns the String representation of the current value.

#### Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicInteger.h`

## 6.121 decaf::util::concurrent::atomic::AtomicRefCounter Class Reference

```
#include <src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h>
```

Inherited by `decaf::lang::ArrayPointer< unsigned char >`, `decaf::lang::Pointer< activemq::threads::TaskRunner >`, `decaf::lang::Pointer< ActiveMQDestination >`, `decaf::lang::Pointer< ActiveMQTransactionContext >`, `decaf::lang::Pointer< BackupTransportPool >`, `decaf::lang::Pointer< BooleanExpression >`, `decaf::lang::Pointer< BrokerError >`, `decaf::lang::Pointer< BrokerId >`, `decaf::lang::Pointer< ByteArrayAdapter >`, `decaf::lang::Pointer< CloseTransportsTask >`, `decaf::lang::Pointer< cms::Destination >`, `decaf::lang::Pointer< commands::ActiveMQDestination >`, `decaf::lang::Pointer< commands::ConsumerId >`, `decaf::lang::Pointer< commands::ConsumerInfo >`, `decaf::lang::Pointer< commands::Message >`, `decaf::lang::Pointer< commands::MessageAck >`, `decaf::lang::Pointer< commands::ProducerInfo >`, `decaf::lang::Pointer< commands::SessionInfo >`, `decaf::lang::Pointer< commands::TransactionId >`, `decaf::lang::Pointer< commands::WireFormatInfo >`, `decaf::lang::Pointer< Comparator< E > >`, `decaf::lang::Pointer< CompositeTaskRunner >`, `decaf::lang::Pointer< ConnectionId >`, `decaf::lang::Pointer< ConnectionInfo >`, `decaf::lang::Pointer< ConsumerId >`, `decaf::lang::Pointer< ConsumerInfo >`, `decaf::lang::Pointer< core::ActiveMQAckHandler >`, `decaf::lang::Pointer< DataStructure >`, `decaf::lang::Pointer< decaf::lang::Runnable >`, `decaf::lang::Pointer< decaf::lang::Thread >`, `decaf::lang::Pointer< Exception >`, `decaf::lang::Pointer< LockHandle >`, `decaf::lang::Pointer< LogManagerInternals >`, `decaf::lang::Pointer< Message >`, `decaf::lang::Pointer< MessageAck >`, `decaf::lang::Pointer< MessageId >`, `decaf::lang::Pointer< ProducerId >`, `decaf::lang::Pointer< ProducerInfo >`, `decaf::lang::Pointer< Properties >`, `decaf::lang::Pointer< Response >`,

decaf::lang::Pointer< ResponseBuilder >, decaf::lang::Pointer< SessionId >, decaf::lang::Pointer< SessionInfo >, decaf::lang::Pointer< Tracked >, decaf::lang::Pointer< TransactionId >, decaf::lang::Pointer< TransactionState >, decaf::lang::Pointer< Transport >, decaf::lang::Pointer< TransportListener >, decaf::lang::Pointer< URI >, decaf::lang::Pointer< URIPool >, and decaf::lang::Pointer< wireformat::WireFormat >.

## Public Member Functions

- **AtomicRefCounter** ()
- **AtomicRefCounter** (const **AtomicRefCounter** &other)
- virtual ~**AtomicRefCounter** ()

## Protected Member Functions

- void **swap** (**AtomicRefCounter** &other)  
*Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.*
- bool **release** ()  
*Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.*

## 6.121.1 Constructor & Destructor Documentation

- 6.121.1.1 **decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter** ( )  
[inline]
- 6.121.1.2 **decaf::util::concurrent::atomic::AtomicRefCounter::AtomicRefCounter** ( const **AtomicRefCounter** & other ) [inline]

References decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet().

- 6.121.1.3 **virtual decaf::util::concurrent::atomic::AtomicRefCounter::~~AtomicRefCounter** ( )  
[inline, virtual]

## 6.121.2 Member Function Documentation

- 6.121.2.1 **bool decaf::util::concurrent::atomic::AtomicRefCounter::release** ( ) [inline, protected]

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.



## Returns

true if the count is now zero.

Reimplemented in `decaf::lang::ArrayPointer< unsigned char >` (p. 703), `decaf::lang::Pointer< MessageAck >` (p. 2902), `decaf::lang::Pointer< BooleanExpression >` (p. 2902), `decaf::lang::Pointer< commands::ConsumerId >` (p. 2902), `decaf::lang::Pointer< BrokerError >` (p. 2902), `decaf::lang::Pointer< Transport >` (p. 2902), `decaf::lang::Pointer< wireformat::WireFormat >` (p. 2902), `decaf::lang::Pointer< commands::WireFormatInfo >` (p. 2902), `decaf::lang::Pointer< CloseTransportsTask >` (p. 2902), `decaf::lang::Pointer< CompositeTaskRunner >` (p. 2902), `decaf::lang::Pointer< ActiveMQTransactionContext >` (p. 2902), `decaf::lang::Pointer< commands::ProducerInfo >` (p. 2902), `decaf::lang::Pointer< Comparator< E > >` (p. 2902), `decaf::lang::Pointer< BrokerId >` (p. 2902), `decaf::lang::Pointer< LogManagerInternals >` (p. 2902), `decaf::lang::Pointer< commands::SessionInfo >` (p. 2902), `decaf::lang::Pointer< Message >` (p. 2902), `decaf::lang::Pointer< URI >` (p. 2902), `decaf::lang::Pointer< DataStructure >` (p. 2902), `decaf::lang::Pointer< activemq::threads::TaskRunner >` (p. 2902), `decaf::lang::Pointer< LockHandle >` (p. 2902), `decaf::lang::Pointer< commands::ActiveMQDestination >` (p. 2902), `decaf::lang::Pointer< ConsumerInfo >` (p. 2902), `decaf::lang::Pointer< ConnectionId >` (p. 2902), `decaf::lang::Pointer< decaf::lang::Runnable >` (p. 2902), `decaf::lang::Pointer< Properties >` (p. 2902), `decaf::lang::Pointer< BackupTransportPool >` (p. 2902), `decaf::lang::Pointer< ProducerInfo >` (p. 2902), `decaf::lang::Pointer< decaf::lang::Thread >` (p. 2902), `decaf::lang::Pointer< Messageld >` (p. 2902), `decaf::lang::Pointer< Response >` (p. 2902), `decaf::lang::Pointer< SessionId >` (p. 2902), `decaf::lang::Pointer< cms::Destination >` (p. 2902), `decaf::lang::Pointer< TransportListener >` (p. 2902), `decaf::lang::Pointer< commands::TransactionId >` (p. 2902), `decaf::lang::Pointer< ActiveMQDestination >` (p. 2902), `decaf::lang::Pointer< ProducerId >` (p. 2902), `decaf::lang::Pointer< ResponseBuilder >` (p. 2902), `decaf::lang::Pointer< SessionInfo >` (p. 2902), `decaf::lang::Pointer< commands::Message >` (p. 2902), `decaf::lang::Pointer< Tracked >` (p. 2902), `decaf::lang::Pointer< ConnectionInfo >` (p. 2902), `decaf::lang::Pointer< commands::MessageAck >` (p. 2902), `decaf::lang::Pointer< core::ActiveMQAckHandler >` (p. 2902), `decaf::lang::Pointer< Exception >` (p. 2902), `decaf::lang::Pointer< TransactionState >` (p. 2902), `decaf::lang::Pointer< commands::ConsumerInfo >` (p. 2902), `decaf::lang::Pointer< ConsumerId >` (p. 2902), `decaf::lang::Pointer< URIPool >` (p. 2902), `decaf::lang::Pointer< ByteArrayAdapter >` (p. 2902), and `decaf::lang::Pointer< TransactionId >` (p. 2902).

References `decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet()`.

**6.121.2.2** `void decaf::util::concurrent::atomic::AtomicRefCounter::swap ( AtomicRefCounter & other ) [inline, protected]`

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

## Parameters

<i>other</i>	The value to swap with this one's.
--------------	------------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicRefCounter.h**

## 6.122 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference

An Pointer reference that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>
```

### Public Member Functions

- **AtomicReference** ()
- **AtomicReference** (T \*value)
- virtual ~**AtomicReference** ()
- T \* **get** () const  
*Gets the Current Value.*
- void **set** (T \*newValue)  
*Sets the Current value of this Reference.*
- bool **compareAndSet** (T \*expect, T \*update)  
*Atomically sets the value to the given updated value if the current value == the expected value.*
- T \* **getAndSet** (T \*newValue)  
*Atomically sets to the given value and returns the old value.*
- std::string **toString** () const  
*Returns the String representation of the current value.*

### 6.122.1 Detailed Description

```
template<typename T>class decaf::util::concurrent::atomic::AtomicReference< T >
```

An Pointer reference that may be updated atomically.

### 6.122.2 Constructor & Destructor Documentation

6.122.2.1 `template<typename T> decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference ( ) [inline]`

6.122.2.2 `template<typename T> decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference ( T * value ) [inline]`

6.122.2.3 `template<typename T> virtual decaf::util::concurrent::atomic::AtomicReference< T >::~~AtomicReference ( ) [inline, virtual]`

### 6.122.3 Member Function Documentation

6.122.3.1 `template<typename T> bool decaf::util::concurrent::atomic::AtomicReference< T >::compareAndSet ( T * expect, T * update )` `[inline]`

Atomically sets the value to the given updated value if the current value == the expected value.

#### Parameters

<i>expect</i>	- the expected value
<i>update</i>	- the new value

#### Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.122.3.2 `template<typename T> T* decaf::util::concurrent::atomic::AtomicReference< T >::get ( ) const` `[inline]`

Gets the Current Value.

#### Returns

the current value of this Reference.

6.122.3.3 `template<typename T> T* decaf::util::concurrent::atomic::AtomicReference< T >::getAndSet ( T * newValue )` `[inline]`

Atomically sets to the given value and returns the old value.

#### Parameters

<i>newValue</i>	- the new value
-----------------	-----------------

#### Returns

the previous value.

6.122.3.4 `template<typename T> void decaf::util::concurrent::atomic::AtomicReference< T >::set ( T * newValue )` `[inline]`

Sets the Current value of this Reference.

#### Parameters

<i>newValue</i>	The new Value of this Reference.
-----------------	----------------------------------

```
6.122.3.5  template<typename T > std::string
           decaf::util::concurrent::atomic::AtomicReference< T
           >::toString ( ) const    [inline]
```

Returns the String representation of the current value.

### Returns

string representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicReference.h**

## 6.123 activemq::transport::failover::BackupTransport Class Reference

```
#include <src/main/activemq/transport/failover/BackupTransport.h>
```

Inheritance diagram for activemq::transport::failover::BackupTransport:

### Public Member Functions

- **BackupTransport** (**BackupTransportPool** \*failover)
- virtual ~**BackupTransport** ()
- **decaf::net::URI** **getUri** () const  
*Gets the URI assigned to this Backup.*
- void **setUri** (const **decaf::net::URI** &uri)  
*Sets the URI assigned to this **Transport** (p. 3819).*
- const **Pointer**< **Transport** > & **getTransport** ()  
*Gets the currently held transport.*
- void **setTransport** (const **Pointer**< **Transport** > &transport)  
*Sets the held transport, if not NULL then start to listen for exceptions from the held transport.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- bool **isClosed** () const  
*Has the **Transport** (p. 3819) been shutdown and no longer usable.*
- void **setClosed** (bool value)  
*Sets the closed flag on this **Transport** (p. 3819).*

### 6.123.1 Constructor & Destructor Documentation

6.123.1.1 `activemq::transport::failover::BackupTransport::BackupTransport ( BackupTransportPool * failover )`

6.123.1.2 `virtual activemq::transport::failover::BackupTransport::~~BackupTransport ( )`  
[virtual]

### 6.123.2 Member Function Documentation

6.123.2.1 `const Pointer<Transport>& activemq::transport::failover::BackupTransport::getTransport ( )`  
[inline]

Gets the currently held transport.

#### Returns

pointer to the held transport or NULL if not set.

6.123.2.2 `decaf::net::URI activemq::transport::failover::BackupTransport::getUri ( ) const`  
[inline]

Gets the URI assigned to this Backup.

#### Returns

the assigned URI

6.123.2.3 `bool activemq::transport::failover::BackupTransport::isClosed ( ) const`  
[inline]

Has the **Transport** (p. 3819) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3819)

6.123.2.4 `virtual void activemq::transport::failover::BackupTransport::onException ( const decaf::lang::Exception & ex )` [virtual]

Event handler for an exception from a command transport.

The **BackupTransport** (p. 718) closes its internal **Transport** (p. 3819) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

#### Parameters

<i>ex</i>	The exception that was passed to this listener to handle.
-----------	---

Implements **activemq::transport::TransportListener** (p. 3837).

6.123.2.5 **void activemq::transport::failover::BackupTransport::setClosed ( bool *value* )**  
[inline]

Sets the closed flag on this **Transport** (p. 3819).

#### Parameters

<i>value</i>	- true for closed.
--------------	--------------------

6.123.2.6 **void activemq::transport::failover::BackupTransport::setTransport ( const Pointer< Transport > & *transport* )** [inline]

Sets the held transport, if not NULL then start to listen for exceptions from the held transport.

#### Parameters

<i>transport</i>	The transport to hold.
------------------	------------------------

6.123.2.7 **void activemq::transport::failover::BackupTransport::setUri ( const decaf::net::URI & *uri* )** [inline]

Sets the URI assigned to this **Transport** (p. 3819).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransport.h**

## 6.124 activemq::transport::failover::BackupTransportPool Class Reference

```
#include <src/main/activemq/transport/failover/BackupTransportPool.h>
```

Inheritance diagram for activemq::transport::failover::BackupTransportPool:

#### Public Member Functions

- **BackupTransportPool** (const Pointer< CompositeTaskRunner > &taskRun-

ner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)

- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual ~**BackupTransportPool** ()
- virtual bool **isPending** () const

*Return true if we don't currently have enough Connected Transports.*

- **Pointer**< **BackupTransport** > **getBackup** ()

*Get a Connected **Transport** (p. 3819) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*

- virtual bool **iterate** ()

*Connect to a Backup Broker if we haven't already connected to the max number of Backups.*

- int **getBackupPoolSize** () const

*Gets the Max number of Backups this Task will create.*

- void **setBackupPoolSize** (int size)

*Sets the Max number of Backups this Task will create.*

- bool **isEnabled** () const

*Gets if the backup **Transport** (p. 3819) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*

- void **setEnabled** (bool value)

*Sets if this Backup **Transport** (p. 3819) Pool is enabled.*

## Friends

- class **BackupTransport**

## 6.124.1 Constructor & Destructor Documentation

6.124.1.1 **activemq::transport::failover::BackupTransportPool::BackupTransportPool** ( const **Pointer**< **CompositeTaskRunner** > & taskRunner, const **Pointer**< **CloseTransportsTask** > & closeTask, const **Pointer**< **URIPool** > & uriPool )

6.124.1.2 **activemq::transport::failover::BackupTransportPool::BackupTransportPool** ( int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > & taskRunner, const **Pointer**< **CloseTransportsTask** > & closeTask, const **Pointer**< **URIPool** > & uriPool )

6.124.1.3 **virtual activemq::transport::failover::BackupTransportPool::~BackupTransportPool** ( ) [virtual]

## 6.124.2 Member Function Documentation

6.124.2.1 **Pointer<BackupTransport>** `activemq::transport::failover::BackupTransportPool::getBackup ( )`

Get a Connected **Transport** (p. 3819) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

#### Returns

Pointer to a Connected **Transport** (p. 3819) or NULL

6.124.2.2 **int** `activemq::transport::failover::BackupTransportPool::getBackupPoolSize ( )` `const [inline]`

Gets the Max number of Backups this Task will create.

#### Returns

the max number of active BackupTransports that will be created.

6.124.2.3 **bool** `activemq::transport::failover::BackupTransportPool::isEnabled ( )` `const [inline]`

Gets if the backup **Transport** (p. 3819) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

#### Returns

true if enable.

6.124.2.4 **virtual bool** `activemq::transport::failover::BackupTransportPool::isPending ( )` `const [virtual]`

Return true if we don't currently have enough Connected Transports.

Implements **activemq::threads::CompositeTask** (p. 1194).

6.124.2.5 **virtual bool** `activemq::transport::failover::BackupTransportPool::iterate ( )` `[virtual]`

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements **activemq::threads::Task** (p. 3679).



6.124.2.6 `void activemq::transport::failover::BackupTransportPool::setBackupPoolSize ( int size ) [inline]`

Sets the Max number of Backups this Task will create.

#### Parameters

<i>size</i>	- the max number of active BackupTransports that will be created.
-------------	---

6.124.2.7 `void activemq::transport::failover::BackupTransportPool::setEnabled ( bool value )`

Sets if this Backup **Transport** (p. 3819) Pool is enabled.

When not enabled no Backups are created and any that were are destroyed.

#### Parameters

<i>value</i>	- true to enable backup creation, false to disable.
--------------	---

### 6.124.3 Friends And Related Function Documentation

6.124.3.1 `friend class BackupTransport [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransportPool.h`

## 6.125 activemq::commands::BaseCommand Class Reference

```
#include <src/main/activemq/commands/BaseCommand.h>
```

Inheritance diagram for `activemq::commands::BaseCommand`:

### Public Member Functions

- **BaseCommand** ()
- virtual **~BaseCommand** ()
- virtual void **setCommandId** (int id)  
*Sets the **Command** (p. 1165) Id of this **Message** (p. 2475).*
- virtual int **getCommandId** () const  
*Gets the **Command** (p. 1165) Id of this **Message** (p. 2475).*
- virtual void **setResponseRequired** (const bool required)  
*Set if this **Message** (p. 2475) requires a **Response** (p. 3227).*

- virtual bool **isResponseRequired** () const  
*Is a **Response** (p. 3227) required for this **Command** (p. 1165).*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

### 6.125.1 Constructor & Destructor Documentation

6.125.1.1 `activemq::commands::BaseCommand::BaseCommand ( )` [inline]

6.125.1.2 `virtual activemq::commands::BaseCommand::~~BaseCommand ( )` [inline, virtual]

### 6.125.2 Member Function Documentation

6.125.2.1 `virtual void activemq::commands::BaseCommand::copyDataStructure ( const DataStructure * src )` [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1629).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 174), **activemq::commands::ActiveMQBytesMessage** (p. 205), **activemq::commands::ActiveMQMapMessage** (p. 334), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 415), **activemq::commands::ActiveMQStreamMessage** (p. 510), **activemq::commands::ActiveMQTextMessage** (p. 633), **activemq::commands::BrokerError** (p. 825), **activemq::commands::BrokerInfo** (p. 858), **activemq::commands::ConnectionControl** (p. 1239), **activemq::commands::ConnectionError** (p. 1267), **activemq::commands::ConnectionInfo** (p. 1327), **activemq::commands::ConsumerControl** (p. 1371), **activemq::commands::ConsumerInfo** (p. 1429), **activemq::commands::ControlCommand** (p. 1461), **activemq::commands::DataArrayResponse** (p. 1494), **activemq::commands::DataResponse** (p. 1551), **activemq::commands::DestinationInfo** (p. 1693), **activemq::commands::ExceptionResponse** (p. 1803), **activemq::commands::FlushCommand** (p. 1902), **activemq::commands::IntegerResponse** (p. 2055), **activemq::commands::KeepAliveInfo** (p. 2227), **activemq::commands::Message** (p. 2481), **activemq::commands::MessageAck** (p. 2522), **activemq::commands::MessageDispatch** (p. 2556), **activemq::commands::MessageDispatchNotification** (p. 2592), **activemq::commands::MessagePull** (p. 2697), **activemq::commands::ProducerAck** (p. 2985), **activemq::commands::ProducerInfo** (p. 3044), **activemq::commands::RemoveInfo** (p. 3139), **activemq::commands::RemoveSubscriptionInfo** (p. 3167), **activemq::commands::ReplayCommand** (p. 3195), **activemq::commands::Response** (p. 3229), **activemq::commands::SessionInfo** (p. 3350), **activemq::commands::ShutdownInfo** (p. 3414), **activemq::commands::TransactionInfo** (p. 3787), and **activemq::commands::WireFormatInfo** (p. 3915).

References `getCommandId()`, and `isResponseRequired()`.

**6.125.2.2** `virtual bool activemq::commands::BaseCommand::equals ( const DataStructure *  
value ) const [inline, virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 175), **activemq::commands::ActiveMQBytesMessage** (p. 206), **activemq::commands::ActiveMQMapMessage** (p. 334), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 399), **activemq::commands::ActiveMQObjectMessage** (p. 415), **activemq::commands::ActiveMQStreamMessage** (p. 510), **activemq::commands::ActiveMQTextMessage** (p. 633), **activemq::commands::BrokerInfo** (p. 858), **activemq::commands::ConnectionControl** (p. 1239), **activemq::commands::ConnectionError** (p. 1268), **activemq::commands::ConnectionInfo** (p. 1327), **activemq::commands::ConsumerControl** (p. 1371), **activemq::commands::ConsumerInfo** (p. 1429), **activemq::commands::ControlCommand** (p. 1461), **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse** (p. 1551), **activemq::commands::DestinationInfo** (p. 1693), **activemq::commands::ExceptionResponse** (p. 1803), **activemq::commands::FlushCommand** (p. 1902), **activemq::commands::IntegerResponse** (p. 2055), **activemq::commands::KeepAliveInfo** (p. 2227), **activemq::commands::Message** (p. 2481), **activemq::commands::MessageAck** (p. 2523), **activemq::commands::MessageDispatch** (p. 2557), **activemq::commands::MessageDispatchNotification**

(p. 2592), **activemq::commands::MessagePull** (p. 2697), **activemq::commands::ProducerAck** (p. 2986), **activemq::commands::ProducerInfo** (p. 3045), **activemq::commands::RemoveInfo** (p. 3139), **activemq::commands::RemoveSubscriptionInfo** (p. 3167), **activemq::commands::ReplayCommand** (p. 3195), **activemq::commands::Response** (p. 3229), **activemq::commands::SessionInfo** (p. 3350), **activemq::commands::ShutdownInfo** (p. 3414), **activemq::commands::TransactionInfo** (p. 3787), **activemq::commands::WireFormatInfo** (p. 3915), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 399), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 399).

References **activemq::commands::BaseDataStructure::equals()**.

**6.125.2.3** `virtual int activemq::commands::BaseCommand::getCommandId ( ) const`  
`[inline, virtual]`

Gets the **Command** (p. 1165) Id of this **Message** (p. 2475).

#### Returns

**Command** (p. 1165) Id

Implements **activemq::commands::Command** (p. 1166).

Referenced by **copyDataStructure()**.

**6.125.2.4** `virtual bool activemq::commands::BaseCommand::isBrokerInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1167).

Reimplemented in **activemq::commands::BrokerInfo** (p. 860).

**6.125.2.5** `virtual bool activemq::commands::BaseCommand::isConnectionInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1167).

Reimplemented in **activemq::commands::ConnectionInfo** (p. 1328).

**6.125.2.6** `virtual bool activemq::commands::BaseCommand::isConsumerInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1167).

Reimplemented in **activemq::commands::ConsumerInfo** (p. 1431).

6.125.2.7 `virtual bool activemq::commands::BaseCommand::isKeepAliveInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1167).

Reimplemented in **activemq::commands::KeepAliveInfo** (p. 2227).

6.125.2.8 `virtual bool activemq::commands::BaseCommand::isMessage ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1167).

Reimplemented in **activemq::commands::Message** (p. 2486).

6.125.2.9 `virtual bool activemq::commands::BaseCommand::isMessageAck ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1167).

Reimplemented in **activemq::commands::MessageAck** (p. 2524).

6.125.2.10 `virtual bool activemq::commands::BaseCommand::isMessageDispatch ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1167).

Reimplemented in **activemq::commands::MessageDispatch** (p. 2558).

6.125.2.11 `virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1168).

Reimplemented in **activemq::commands::MessageDispatchNotification** (p. 2593).

6.125.2.12 `virtual bool activemq::commands::BaseCommand::isProducerAck ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1168).

Reimplemented in **activemq::commands::ProducerAck** (p. 2986).

6.125.2.13 `virtual bool activemq::commands::BaseCommand::isProducerInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1168).

Reimplemented in **activemq::commands::ProducerInfo** (p. 3046).

6.125.2.14 `virtual bool activemq::commands::BaseCommand::isRemoveInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1168).

Reimplemented in **activemq::commands::RemoveInfo** (p. 3140).

6.125.2.15 `virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo ( )`  
`const [inline, virtual]`

Implements **activemq::commands::Command** (p. 1168).

Reimplemented in **activemq::commands::RemoveSubscriptionInfo** (p. 3168).

6.125.2.16 `virtual bool activemq::commands::BaseCommand::isResponse ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1168).

Reimplemented in **activemq::commands::Response** (p. 3230).

6.125.2.17 `virtual bool activemq::commands::BaseCommand::isResponseRequired ( ) const`  
`[inline, virtual]`

Is a **Response** (p. 3227) required for this **Command** (p. 1165).

### Returns

true if a response is required.

Implements **activemq::commands::Command** (p. 1168).

Referenced by `copyDataStructure()`.

6.125.2.18 `virtual bool activemq::commands::BaseCommand::isShutdownInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1169).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 3415).

6.125.2.19 `virtual bool activemq::commands::BaseCommand::isTransactionInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1169).

Reimplemented in **activemq::commands::TransactionInfo** (p. 3788).

6.125.2.20 `virtual bool activemq::commands::BaseCommand::isWireFormatInfo ( ) const`  
`[inline, virtual]`

Implements **activemq::commands::Command** (p. 1169).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 3919).

6.125.2.21 `virtual void activemq::commands::BaseCommand::setCommandId ( int id )`  
`[inline, virtual]`

Sets the **Command** (p. 1165) Id of this **Message** (p. 2475).

#### Parameters

<i>id</i>	<b>Command</b> (p. 1165) Id
-----------	-----------------------------

Implements **activemq::commands::Command** (p. 1169).

6.125.2.22 `virtual void activemq::commands::BaseCommand::setResponseRequired ( const`  
`bool required ) [inline, virtual]`

Set if this **Message** (p. 2475) requires a **Response** (p. 3227).

#### Parameters

<i>required</i>	true if response is required
-----------------	------------------------------

Implements **activemq::commands::Command** (p. 1169).

6.125.2.23 `virtual std::string activemq::commands::BaseCommand::toString ( ) const`  
`[inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Implements **activemq::commands::Command** (p. 1169).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 177), **activemq::commands::ActiveMQBytesMessage** (p. 214), **activemq::commands::ActiveMQMapMessage** (p. 344), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 416), **activemq::commands::ActiveMQStreamMessage** (p. 518), **activemq::commands::ActiveMQTextMessage** (p. 635), **activemq::commands::BrokerInfo** (p. 861), **activemq::commands::ConnectionControl** (p. 1241), **activemq::commands::ConnectionError** (p. 1269), **activemq::commands::ConnectionInfo** (p. 1329), **activemq::commands::ConsumerControl** (p. 1372), **activemq::commands::ConsumerInfo** (p. 1432), **activemq::commands::ControlCommand** (p. 1462), **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse**

(p. 1552), **activemq::commands::DestinationInfo** (p. 1695), **activemq::commands::ExceptionResponse** (p. 1804), **activemq::commands::FlushCommand** (p. 1902), **activemq::commands::IntegerResponse** (p. 2056), **activemq::commands::KeepAliveInfo** (p. 2228), **activemq::commands::Message** (p. 2490), **activemq::commands::MessageAck** (p. 2525), **activemq::commands::MessageDispatch** (p. 2558), **activemq::commands::MessageDispatchNotification** (p. 2594), **activemq::commands::MessageInfo** (p. 2699), **activemq::commands::ProducerAck** (p. 2987), **activemq::commands::ProducerInfo** (p. 3046), **activemq::commands::RemoveInfo** (p. 3140), **activemq::commands::RemoveSubscriptionInfo** (p. 3168), **activemq::commands::ReplayCommand** (p. 3196), **activemq::commands::Response** (p. 3230), **activemq::commands::SessionInfo** (p. 3351), **activemq::commands::ShutdownInfo** (p. 3415), **activemq::commands::TransactionInfo** (p. 3788), and **activemq::commands::WireFormatInfo** (p. 3922).

References **activemq::commands::BaseDataStructure::toString()**.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseCommand.h`

## 6.126 **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 730).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )



*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.126.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 730).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.126.2 Constructor & Destructor Documentation

6.126.2.1 **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::BaseCommandMarshaller**  
 ( ) [inline]

6.126.2.2 **virtual activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::~~BaseCommandMarshaller**  
 ( ) [inline, virtual]

## 6.126.3 Member Function Documentation

6.126.3.1 **virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseMarshal**  
 ( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 179), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 222), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 346), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 373), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 418), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

(p. 525), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 637), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 864), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1244), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1275), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1336), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1379), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1440), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1468), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1567), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1702), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1814), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1913), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2066), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2238), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2536), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2572), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2605), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2658), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2709), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2997), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3065), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3151), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3175), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3252), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3365), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3434), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3799).

```
6.126.3.2  virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1599).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 180), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 222), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 419), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 525), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 638), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 865), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1244), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1276), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1337), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1380), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1441), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1469), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1567), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1702), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1815), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1913), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2067), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2239), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2536), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2572), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2605), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2659), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2710), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2998), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3066), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3151), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3176), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3253), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3366), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3434), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3799).

```
6.126.3.3  virtual int activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 180), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 347), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 373), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 419), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 526), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 638), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 865), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1244), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1276), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1337), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1380), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1441), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1469), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1503), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1568), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1702), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1815), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1914), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2067), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2239), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 2537), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2573), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 2606), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2659), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2710), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2998), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 3066), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 3152), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 3176), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 3211), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 3366), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3434), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3800).

6.126.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 181), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 347), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 374), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 420), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 526), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 639), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1245), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1277), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1338), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1381), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1442), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1470), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1503), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1568), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1816), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1914), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2068), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2240), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 2537), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2573), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 2606), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2660), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2711), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2999), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 3067), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 3152), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 3177),

**activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 3212), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 3367), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3435), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3800).

```
6.126.3.5 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 181), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 224), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 348), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 374), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 420), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 527), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 639), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 866), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 1245), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 1277), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1338), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1381), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1442), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1470), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1504), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1569), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1703), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1816), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1915), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2068), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2240), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 2538), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2574),

`activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2607), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2661), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2711), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2999), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3067), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3153), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3177), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3212), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3254), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 3367), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3435), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3801).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h`

## 6.127 `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 737).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`:

### Public Member Functions

- `BaseCommandMarshaller ()`
- `virtual ~BaseCommandMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.127.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 737).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.127.2 Constructor & Destructor Documentation

6.127.2.1 **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.127.2.2 **virtual activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::~~BaseCommandMarshaller**  
( ) [inline, virtual]

### 6.127.3 Member Function Documentation

6.127.3.1 **virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \*  
dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 230), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 354), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 381), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 426), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**



(p. 533), [activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller](#) (p. 641), [activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller](#) (p. 868), [activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller](#) (p. 1248), [activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller](#) (p. 1279), [activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller](#) (p. 1340), [activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller](#) (p. 1383), [activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller](#) (p. 1444), [activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller](#) (p. 1472), [activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller](#) (p. 1506), [activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller](#) (p. 1571), [activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller](#) (p. 1706), [activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller](#) (p. 1822), [activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller](#) (p. 1917), [activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller](#) (p. 2070), [activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller](#) (p. 2242), [activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller](#) (p. 2540), [activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller](#) (p. 2580), [activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller](#) (p. 2609), [activemq::wireformat::openwire::marshal::v4::MessageMarshaller](#) (p. 2667), [activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller](#) (p. 2713), [activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller](#) (p. 2993), [activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller](#) (p. 3049), [activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller](#) (p. 3163), [activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller](#) (p. 3191), [activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller](#) (p. 3199), [activemq::wireformat::openwire::marshal::v4::ResponseMarshaller](#) (p. 3238), [activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller](#) (p. 3373), [activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller](#) (p. 3438), and [activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller](#) (p. 3807).

```
6.127.3.2  virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
            [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1599).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 381), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 427), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 642), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 869), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1248), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1341), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1384), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1445), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1473), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1506), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1571), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1706), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1823), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1917), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2071), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2243), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2540), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2580), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2609), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2668), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2714), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2994), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3050), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3163), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3192), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3199), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3239), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3374), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3438), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3807).

```
6.127.3.3  virtual int activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1606).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 355), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 381), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 427), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 534), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 642), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 869), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1248), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1341), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1384), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1445), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1473), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1507), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1572), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1706), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1823), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1918), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2071), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2243), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2541), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2581), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2610), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2668), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2714), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2994), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3050), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3164), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3192), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3199), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3239), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3374), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3438), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3808).

```
6.127.3.4 virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 231), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 355), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 382), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 428), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 534), **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 643), **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** (p. 870), **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller** (p. 1249), **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller** (p. 1281), **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller** (p. 1342), **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller** (p. 1385), **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller** (p. 1446), **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller** (p. 1474), **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1507), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1572), **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller** (p. 1707), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1824), **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller** (p. 1918), **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2072), **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller** (p. 2244), **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller** (p. 2541), **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller** (p. 2581), **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller** (p. 2610), **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669), **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller** (p. 2715), **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller** (p. 2995), **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller** (p. 3051), **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller** (p. 3164), **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller** (p. 3193),

`activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3200), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3240), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 3375), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3439), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3808).

6.127.3.5 `virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1620).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 189), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 232), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 356), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 382), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 428), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 535), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 643), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 870), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1249), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1281), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1342), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1385), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1446), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1474), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1508), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1573), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1707), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1824), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1919), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2072), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2244), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2542), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2582),

**activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller** (p. 2611), **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669), **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller** (p. 2715), **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller** (p. 2995), **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller** (p. 3051), **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller** (p. 3165), **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller** (p. 3193), **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller** (p. 3200), **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240), **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller** (p. 3375), **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller** (p. 3439), and **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller** (p. 3809).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h`

## 6.128 **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 743).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.128.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 743).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.128.2 Constructor & Destructor Documentation

6.128.2.1 **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.128.2.2 **virtual activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::~~BaseCommandMarshaller**  
( ) [inline, virtual]

## 6.128.3 Member Function Documentation

6.128.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 226), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 350), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 422), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

(p. 529), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 645), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 872), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1252), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1283), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1344), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1387), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1448), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1476), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1510), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1575), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1710), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1826), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1921), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2074), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2250), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2544), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2584), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2613), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2671), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2717), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3009), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3057), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3155), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3171), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3203), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3257), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3361), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3426), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3795).

```
6.128.3.2  virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1599).



Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 226), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 423), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 529), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 646), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 873), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1252), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1284), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1345), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1388), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1449), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1477), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1510), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1575), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1710), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1827), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1921), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2075), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2251), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2544), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2584), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2614), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2672), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2718), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3010), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3058), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3155), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3172), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3203), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3257), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 3362), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3426), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3795).

```
6.128.3.3  virtual int activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 530), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 646), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 873), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1252), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1284), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1345), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1388), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1449), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1477), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1511), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1576), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1710), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1827), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1922), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2075), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2251), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2545), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2585), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 2614), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2672), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 2718), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 3010), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 3058), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 3156), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 3172), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 3203), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 3362), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 3426), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3796).

6.128.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 378), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 424), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 530), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 647), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 874), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1253), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1285), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1346), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1389), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1450), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1478), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1511), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1576), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1711), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1828), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1922), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2076), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2252), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2545), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2585), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 2615), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2673), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 2719), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 3011), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 3059), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 3156), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 3173),

**activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 3204), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 3363), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 3427), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3796).

```
6.128.3.5 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 352), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 378), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 424), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 531), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 647), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 874), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1253), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1285), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1346), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1389), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1450), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1478), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1512), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1577), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1711), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1828), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1923), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2076), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2252), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2546), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2586),

activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 2615), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2674), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2719), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3011), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3059), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3157), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3173), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3204), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3259), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 3363), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3427), and activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3797).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h

## 6.129 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 750).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.129.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 750).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.129.2 Constructor & Destructor Documentation

6.129.2.1 **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.129.2.2 **virtual activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::~~BaseCommandMarshaller**  
( ) [inline, virtual]

### 6.129.3 Member Function Documentation

6.129.3.1 **virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \*  
dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 195), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 234), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 358), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 385), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 430), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**

(p. 537), [activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller](#) (p. 649), [activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller](#) (p. 876), [activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller](#) (p. 1256), [activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller](#) (p. 1287), [activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller](#) (p. 1348), [activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller](#) (p. 1391), [activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller](#) (p. 1452), [activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller](#) (p. 1480), [activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller](#) (p. 1514), [activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller](#) (p. 1554), [activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller](#) (p. 1718), [activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller](#) (p. 1818), [activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller](#) (p. 1925), [activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller](#) (p. 2078), [activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller](#) (p. 2246), [activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller](#) (p. 2548), [activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller](#) (p. 2576), [activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller](#) (p. 2617), [activemq::wireformat::openwire::marshal::v5::MessageMarshaller](#) (p. 2654), [activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller](#) (p. 2705), [activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller](#) (p. 3001), [activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller](#) (p. 3061), [activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller](#) (p. 3159), [activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller](#) (p. 3187), [activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller](#) (p. 3219), [activemq::wireformat::openwire::marshal::v5::ResponseMarshaller](#) (p. 3247), [activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller](#) (p. 3357), [activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller](#) (p. 3430), and [activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller](#) (p. 3791).

```
6.129.3.2  virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1599).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 234), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 431), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 537), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 650), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 877), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1256), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1288), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1349), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1392), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1453), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1481), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1514), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1555), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1718), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1819), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1925), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2079), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2247), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2548), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2576), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2618), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2655), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2706), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3002), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3062), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3159), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3188), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3219), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3248), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3358), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3430), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3791).

```
6.129.3.3  virtual int activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1606).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 235), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 359), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 431), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 538), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 650), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 877), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1256), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1288), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1349), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1392), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1453), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1481), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1515), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1555), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1718), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1819), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1926), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2079), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2247), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2549), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2577), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2618), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2655), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2706), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3002), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3062), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3160), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3188), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3219), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3248), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 3358), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3430), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3792).

```
6.129.3.4 virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 197), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 235), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 359), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 386), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 432), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 538), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 651), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 878), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1257), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1289), **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller** (p. 1350), **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller** (p. 1393), **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** (p. 1482), **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1515), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1556), **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller** (p. 1719), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1820), **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller** (p. 1926), **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2080), **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller** (p. 2248), **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller** (p. 2549), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2577), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2619), **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656), **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller** (p. 2707), **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller** (p. 3003), **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller** (p. 3063), **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller** (p. 3160), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller** (p. 3189),

**activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 3220), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3249), **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** (p. 3359), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3431), and **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3792).

6.129.3.5 virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightUnmarshal ( *OpenWireFormat* \* *wireFormat*, *commands::DataStructure* \* *dataStructure*, *decaf::io::DataInputStream* \* *dataIn*, *utils::BooleanStream* \* *bs* ) throw ( *decaf::io::IOException* ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 197), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 360), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 386), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 432), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 539), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 651), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 878), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1257), **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller** (p. 1289), **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller** (p. 1350), **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller** (p. 1393), **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** (p. 1482), **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1516), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1556), **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller** (p. 1719), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1820), **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller** (p. 1927), **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2080), **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller** (p. 2248), **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller** (p. 2550), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2578),

**activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2619), **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2656), **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller** (p. 2707), **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller** (p. 3003), **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller** (p. 3063), **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller** (p. 3161), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller** (p. 3189), **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 3220), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3250), **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller** (p. 3359), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3431), and **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3793).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h`

## 6.130 **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 757).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshall an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.130.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 757).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.130.2 Constructor & Destructor Documentation

6.130.2.1 **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.130.2.2 **virtual activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::~~BaseCommandMarshaller**  
( ) [inline, virtual]

### 6.130.3 Member Function Documentation

6.130.3.1 **virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 199), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 238), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 366), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 438), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**

(p. 545), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 653), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 880), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1260), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1291), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1352), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1395), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1456), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1484), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1518), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1559), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1714), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1806), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1905), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2058), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2230), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2528), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2588), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2597), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2676), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2721), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3005), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3069), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3147), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3183), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3215), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3261), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3353), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3418), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3803).

```
6.130.3.2  virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1599).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 238), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 439), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 545), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 654), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 881), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1260), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1292), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1353), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1396), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1457), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1518), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1559), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1714), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1807), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1905), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2059), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2231), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2528), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2588), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2597), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2676), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2722), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3006), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3070), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3147), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3184), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3215), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3262), `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller` (p. 3354), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3418), and `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3803).

```
6.130.3.3  virtual int activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 200), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 239), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 367), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 439), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 546), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 654), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 881), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1260), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1292), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1353), **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1396), **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1457), **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1485), **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1519), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1714), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1807), **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 1906), **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2059), **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2231), **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2529), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2589), **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller** (p. 2597), **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677), **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller** (p. 2722), **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** (p. 3006), **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller** (p. 3070), **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller** (p. 3148), **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** (p. 3184), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3215), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262), **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3354), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3418), and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3804).



6.130.3.4 virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 239), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 367), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 440), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 546), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 655), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 882), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1261), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1293), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1354), **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1397), **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1458), **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1486), **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1519), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1560), **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1715), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1808), **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 1906), **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2060), **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2232), **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2529), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2589), **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller** (p. 2598), **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2677), **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller** (p. 2723), **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** (p. 3007), **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller** (p. 3071), **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller** (p. 3148), **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** (p. 3185),

**activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3216), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3263), **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller** (p. 3355), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3419), and **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3804).

```
6.130.3.5 virtual void activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 240), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 368), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 440), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 547), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 655), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 882), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1261), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1293), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1354), **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1397), **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1458), **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1486), **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1520), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1560), **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1715), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1808), **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 1907), **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2060), **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2232), **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2530), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2590),

activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller (p. 2598), activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2678), activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2723), activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3007), activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3071), activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3149), activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller (p. 3185), activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller (p. 3216), activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3264), activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller (p. 3355), activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller (p. 3419), and activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3805).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h

## 6.131 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 764).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller:

### Public Member Functions

- **BaseCommandMarshaller** ()
- virtual ~**BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.131.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 764).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.131.2 Constructor & Destructor Documentation

6.131.2.1 **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::BaseCommandMarshaller**  
( ) [inline]

6.131.2.2 **virtual activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::~~BaseCommandMarshaller**  
( ) [inline, virtual]

### 6.131.3 Member Function Documentation

6.131.3.1 **virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseMarshal**  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \*  
dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 191), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 242), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 362), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 434), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

(p. 541), [activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller](#) (p. 657), [activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller](#) (p. 884), [activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller](#) (p. 1264), [activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller](#) (p. 1271), [activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller](#) (p. 1332), [activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller](#) (p. 1375), [activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller](#) (p. 1436), [activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller](#) (p. 1464), [activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller](#) (p. 1498), [activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller](#) (p. 1563), [activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller](#) (p. 1698), [activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller](#) (p. 1810), [activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller](#) (p. 1909), [activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller](#) (p. 2062), [activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller](#) (p. 2234), [activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller](#) (p. 2532), [activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller](#) (p. 2568), [activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller](#) (p. 2601), [activemq::wireformat::openwire::marshal::v2::MessageMarshaller](#) (p. 2663), [activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller](#) (p. 2701), [activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller](#) (p. 2989), [activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller](#) (p. 3053), [activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller](#) (p. 3143), [activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller](#) (p. 3179), [activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller](#) (p. 3207), [activemq::wireformat::openwire::marshal::v2::ResponseMarshaller](#) (p. 3243), [activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller](#) (p. 3369), [activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller](#) (p. 3422), and [activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller](#) (p. 3811).

```
6.131.3.2  virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseUnmarshal
           ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
             decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
           [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements [activemq::wireformat::openwire::marshal::DataStreamMarshaller](#) (p. 1599).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 242), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 435), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 541), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 658), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 885), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1272), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1333), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1376), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1437), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1465), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1563), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1698), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1811), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1909), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2063), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2235), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2532), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2568), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2601), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2663), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2702), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2990), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3054), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3143), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3180), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3207), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3243), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3370), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3422), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3811).

```
6.131.3.3  virtual int activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1606).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 243), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 363), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 435), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 542), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 658), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 885), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1272), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1333), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1376), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1437), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1465), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1564), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1698), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1811), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1910), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2063), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2235), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2533), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2569), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2602), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2664), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2702), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2990), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3054), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3144), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3180), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3207), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3244), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3370), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3422), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3812).

```
6.131.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 193), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 243), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 363), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 390), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 436), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 542), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 659), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 886), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1265), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1273), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1334), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1377), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1438), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1466), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1499), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1564), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1699), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1812), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1910), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2064), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 2236), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 2533), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 2569), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 2602), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 2703), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2991), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 3055), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 3144), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 3181),



**activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 3208), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 3371), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 3423), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 3812).

6.131.3.5 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 193), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 244), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 364), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 390), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 436), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 543), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 659), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 886), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1265), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1273), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1334), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1377), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1438), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1466), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1499), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1565), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1699), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1812), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1911), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2064), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 2236), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 2534), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 2570),

`activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2603), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2665), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2703), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2991), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3055), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3145), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3181), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3208), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3245), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 3371), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3423), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3813).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h`

## 6.132 `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller` Class Reference

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

Inherits `activemq::wireformat::openwire::marshal::DataStreamMarshaller`.

Inherited by `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`, `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`, `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller`, `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller`, `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller`, `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller`, `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller`, `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller`, `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`, `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`, `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller`, `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller`, `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller`, `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller`, `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`, `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller`, `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller`, `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller`, `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`, `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`, `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller`, `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller`, `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller`, `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller`, `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller`, `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller`, `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller`, `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller`, and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller`.

## Public Member Functions

- Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

*Tight Marshal to the given stream.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*format AMQCPP\_UNUSED, **commands::DataStructure** \*command AMQCPP\_UNUSED, **decaf::io::DataInputStream** \*dis AMQCPP\_UNUSED, **utils::BooleanStream** \*bs AMQCPP\_UNUSED) throw ( decaf::io::IOException )

*Tight Un-Marshal to the given stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*format AMQCPP\_UNUSED, **commands::DataStructure** \*command AMQCPP\_UNUSED, **decaf::io::DataOutputStream** \*ds AMQCPP\_UNUSED) throw ( decaf::io::IOException )

*Tight Marshal to the given stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*format AMQCPP\_UNUSED, **commands::DataStructure** \*command AMQCPP\_UNUSED, **decaf::io::DataInputStream** \*dis AMQCPP\_UNUSED) throw ( decaf::io::IOException )

*Loose Un-Marshal to the given stream.*

### Static Public Member Functions

- static std::string **toString** (const **commands::Messageld** \*id)  
*Converts the object to a String.*
- static std::string **toString** (const **commands::ProducerId** \*id)  
*Converts the object to a String.*
- static std::string **toString** (const **commands::TransactionId** \*txnId)  
*Converts the given transaction ID into a String.*
- static std::string **toHexFromBytes** (const std::vector< unsigned char > &data)  
*given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.*

### Protected Member Functions

- virtual **commands::DataStructure** \* **tightUnmarshalCachedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tight Unmarshal the cached object.*
- virtual int **tightMarshalCachedObject1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.*
- virtual void **tightMarshalCachedObject2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Tightly marshals the passed DataStructure based object to the passed streams returning nothing.*
- virtual void **looseMarshalCachedObject** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Loosely marshals the passed DataStructure based object to the passed stream returning nothing.*

- virtual **commands::DataStructure** \* **looseUnmarshalCachedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Loose Unmarshal the cached object.*

- virtual int **tightMarshalNestedObject1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*object, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.*

- virtual void **tightMarshalNestedObject2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*object, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tightly marshals the passed DataStructure based object to the passed streams returning nothing.*

- virtual **commands::DataStructure** \* **tightUnmarshalNestedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tight Unmarshal the nested object.*

- virtual **commands::DataStructure** \* **looseUnmarshalNestedObject** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Loose Unmarshal the nested object.*

- virtual void **looseMarshalNestedObject** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*object, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Loose marshal the nested object.*

- virtual std::string **tightUnmarshalString** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Performs Tight Unmarshaling of String Objects.*

- virtual int **tightMarshalString1** (const std::string &value, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.*

- virtual void **tightMarshalString2** (const std::string &value, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tight Marshals the passed string to the streams passed.*

- virtual void **looseMarshalString** (const std::string value, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Loose Marshal the String to the DataOutputStream passed.*

- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Loose Un-Marshal the String to the DataOutputStream passed.*

- virtual int **tightMarshalLong1** (**OpenWireFormat** \*wireFormat, long long value, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Tightly marshal the long long to the BooleanStream passed.*

- virtual void **tightMarshalLong2** (**OpenWireFormat** \*wireFormat, long long value, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tightly marshal the long long to the Streams passed.*
- virtual long long **tightUnmarshalLong** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight marshal the long long type.*
- virtual void **looseMarshalLong** (**OpenWireFormat** \*wireFormat, long long value, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Tightly marshal the long long to the BooleanStream passed.*
- virtual long long **looseUnmarshalLong** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Loose marshal the long long type.*
- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Unmarshal an array of char.*
- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Loose Unmarshal an array of char.*
- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs, int size) throw ( **decaf::io::IOException** )  
*Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.*
- virtual std::vector< unsigned char > **looseUnmarshalConstByteArray** (**decaf::io::DataInputStream** \*dataIn, int size) throw ( **decaf::io::IOException** )  
*Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.*
- virtual **commands::DataStructure** \* **tightUnmarshalBrokerError** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Unmarshal the Error object.*
- virtual int **tightMarshalBrokerError1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Marshal the Error object.*
- virtual void **tightMarshalBrokerError2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Tight Marshal the Error object.*
- virtual **commands::DataStructure** \* **looseUnmarshalBrokerError** (**OpenWireFormat** \*wireFormat, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Loose Unmarshal the Error object.*
- virtual void **looseMarshalBrokerError** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*data, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Tight Marshal the Error object.*

- template<typename T >  
int **tightMarshalObjectArray1** (OpenWireFormat \*wireFormat, std::vector< T > objects, utils::BooleanStream \*bs) throw ( decaf::io::IOException )  
*Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.*
- template<typename T >  
void **tightMarshalObjectArray2** (OpenWireFormat \*wireFormat, std::vector< T > objects, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )  
*Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.*
- template<typename T >  
void **looseMarshalObjectArray** (OpenWireFormat \*wireFormat, std::vector< T > objects, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )  
*Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.*
- virtual std::string **readAsciiString** (decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )  
*Given an DataInputStream read a know ASCII formatted string from the input and return that string.*

### 6.132.1 Detailed Description

Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.

Since

2.0

### 6.132.2 Constructor & Destructor Documentation

6.132.2.1 virtual activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamMarshaller ( ) [inline, virtual]

### 6.132.3 Member Function Documentation

6.132.3.1 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal ( OpenWireFormat \*format AMQCPP\_UNUSED, commands::DataStructure \*command AMQCPP\_UNUSED, decaf::io::DataOutputStream \*ds AMQCPP\_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]

Tight Marshal to the given stream.

### Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.2 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerError ( OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
`[protected, virtual]`

Tight Marshal the Error object.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- Error to Marshal
<i>dataOut</i>	- stream to write marshalled data to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.3 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedObject ( OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`  
`[protected, virtual]`

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>dataOut</i>	- stream to write marshaled data to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------



```
6.132.3.4 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong
( OpenWireFormat * wireFormat, long long value,
  decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )
[protected, virtual]
```

Tightly marshal the long long to the BooleanStream passed.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>value</i>	- long long to marshal
<i>dataOut</i>	- DataOutputStream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.5 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedObject
( OpenWireFormat * wireFormat, commands::DataStructure * object,
  decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )
[protected, virtual]
```

Loose marshal the nested object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>dataOut</i>	- stream to write marshaled data to

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.6 template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray
( OpenWireFormat * wireFormat, std::vector< T > objects, de-
  caf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )
[inline, protected]
```

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>dataOut</i>	- stream to write marshalled data to

**Returns**

size of the marshalled data

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

```
6.132.3.7 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString
( const std::string value, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [protected, virtual]
```

Loose Marshal the String to the DataOutputStream passed.

**Parameters**

<i>value</i>	- string to marshal
<i>dataOut</i>	- stream to write marshaled form to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.8 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal
( OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure
  *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis
  AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline,
  virtual]
```

Loose Un-Marshal to the given stream.

**Parameters**

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.9 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBrokerError
( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn )
throw ( decaf::io::IOException ) [protected, virtual]
```

Loose Unmarshal the Error object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshalled form from

#### Returns

pointer to a new DataStructure Object

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.10 virtual std::vector<unsigned char> ac-
tivismq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalByteArray
( decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[protected, virtual]
```

Loose Unmarshal an array of char.

#### Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshall from
---------------	---

#### Returns

the unmarshalled vector of chars.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.11 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCachedObject
( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn )
throw ( decaf::io::IOException ) [protected, virtual]
```

Loose Unmarshal the cached object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

**Returns**

pointer to a new DataStructure Object

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.12 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConstByteArray ( decaf::io::DataInputStream * dataIn, int size ) throw ( decaf::io::IOException )` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

**Parameters**

<i>dataIn</i>	- the DataInputStream to Un-Marshall from
<i>size</i>	- size of the const array to unmarshal

**Returns**

the unmarshaled vector of chars.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.13 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong ( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [protected, virtual]

Loose marshal the long long type.

**Parameters**

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

**Returns**

the unmarshaled long long

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.14 virtual **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNestedObject**  
 ( **OpenWireFormat** \* *wireFormat*, **decaf::io::DataInputStream** \* *dataIn* )  
 throw ( **decaf::io::IOException** ) [protected, virtual]

Loose Unmarshal the nested object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from

#### Returns

pointer to a new DataStructure Object

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.15 virtual **std::string** **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString**  
 ( **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
 [protected, virtual]

Loose Un-Marshal the String to the DataOutputStream passed.

#### Parameters

<i>dataIn</i>	- stream to read marshaled form from
---------------	--------------------------------------

#### Returns

the unmarshaled string

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.16 virtual **std::string** **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString**  
 ( **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
 [protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

#### Parameters

<i>dataIn</i>	- DataInputStream to read from
---------------	--------------------------------

**Returns**

string value read from stream

```
6.132.3.17  virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1
( OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure
 *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED )
throw ( decaf::io::IOException ) [inline, virtual]
```

Tight Marshal to the given stream.

**Parameters**

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>bs</i>	- boolean stream to marshal to.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.18  virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2
( OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure
 *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds
 AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED ) throw (
 decaf::io::IOException ) [inline, virtual]
```

Tight Marshal to the given stream.

**Parameters**

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.19  virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError1
( OpenWireFormat * wireFormat, commands::DataStructure *
 data, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[protected, virtual]
```

Tight Marshal the Error object.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- Error to Marshal
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of the marshalled data

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.20  virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerError2
( OpenWireFormat * wireFormat, commands::DataStructure * data,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
  throw ( decaf::io::IOException ) [protected, virtual]
```

Tight Marshal the Error object.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- Error to Marshal
<i>dataOut</i>	- stream to write marshalled data to
<i>bs</i>	- boolean stream to marshal to.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.21  virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedObject1
( OpenWireFormat * wireFormat, commands::DataStructure *
  data, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
  [protected, virtual]
```

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of data written.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.22 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedObject2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *data*,  
 decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* )  
 throw ( decaf::io::IOException ) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

**Parameters**

<i>wireFormat</i>	- The OpenWireFormat properties
<i>data</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.
<i>dataOut</i>	- stream to write marshaled data to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.23 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1  
 ( OpenWireFormat \* *wireFormat*, long long *value*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

**Parameters**

<i>wireFormat</i>	- The OpenWireFormat properties
<i>value</i>	- long long to marshal
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of data written.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------



```
6.132.3.24  virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2
            ( OpenWireFormat * wireFormat, long long value,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
              throw ( decaf::io::IOException ) [protected, virtual]
```

Tightly marshal the long long to the Streams passed.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>value</i>	- long long to marshal
<i>dataOut</i>	- stream to write marshaled form to
<i>bs</i>	- boolean stream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.25  virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              object, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
              [protected, virtual]
```

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.

#### Returns

size of data written.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.26  virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedObject2
            ( OpenWireFormat * wireFormat, commands::DataStructure * object,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
              throw ( decaf::io::IOException ) [protected, virtual]
```

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>object</i>	- DataStructure Object Pointer to marshal
<i>bs</i>	- boolean stream to marshal to.
<i>dataOut</i>	- stream to write marshaled data to

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.27  template<typename T > int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2
( OpenWireFormat * wireFormat, std::vector< T > objects,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [inline,
protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of the marshalled data

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

```
6.132.3.28  template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2
( OpenWireFormat * wireFormat, std::vector< T > objects,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
throw ( decaf::io::IOException ) [inline, protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

**Parameters**

<i>wireFormat</i>	- The OpenwireFormat properties
<i>objects</i>	- array of DataStructure object pointers.
<i>dataOut</i>	- stream to write marshaled data to
<i>bs</i>	- boolean stream to marshal to.

**Returns**

size of the marshalled data

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

References AMQ\_CATCH\_EXCEPTION\_CONVERT, AMQ\_CATCH\_RETHROW, and AMQ\_CATCHALL\_THROW.

```
6.132.3.29  virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1
( const std::string & value, utils::BooleanStream * bs ) throw (
    decaf::io::IOException ) [protected, virtual]
```

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

**Parameters**

<i>value</i>	- string to marshal
<i>bs</i>	- BooleanStream to use.

**Returns**

size of marshaled string.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.30  virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2
( const std::string & value, decaf::io::DataOutputStream * dataOut,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[protected, virtual]
```

Tight Marshals the passed string to the streams passed.

**Parameters**

<i>value</i>	- string to marshal
<i>dataOut</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```

6.132.3.31 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal
( OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure
 *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis
 AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED ) throw (
 decaf::io::IOException ) [inline, virtual]

```

Tight Un-Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to Un-Marshal from.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```

6.132.3.32 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBrokerError
( OpenWireFormat * wireFormat, decaf::io::DataInputStream *
 dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
 [protected, virtual]

```

Tight Unarshall the Error object.

#### Parameters

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshalled form from
<i>bs</i>	- boolean stream to marshal to.

#### Returns

pointer to a new DataStructure Object

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```

6.132.3.33 virtual std::vector<unsigned char> ac-
tivismq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByteArray
( decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
 decaf::io::IOException ) [protected, virtual]

```

Tight Unmarshal an array of char.

**Parameters**

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

**Returns**

the unmarshaled vector of chars.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.34 virtual commands::DataStructure* ac-
    tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCachedObject
    ( OpenWireFormat * wireFormat, decaf::io::DataInputStream *
      dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
    [protected, virtual]
```

Tight Unmarshal the cached object.

**Parameters**

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

**Returns**

pointer to a new DataStructure Object

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.35 virtual std::vector<unsigned char> ac-
    tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConstByteArray
    ( decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs, int size
      ) throw ( decaf::io::IOException ) [protected, virtual]
```

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

**Parameters**

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.
<i>size</i>	- size of the const array to unmarshal

**Returns**

the unmarshaled vector of chars.

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.36 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong ( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
`[protected, virtual]`

Tight marshal the long long type.

**Parameters**

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

**Returns**

the unmarshaled long long

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.132.3.37 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNestedObject ( OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
`[protected, virtual]`

Tight Unmarshal the nested object.

**Parameters**

<i>wireFormat</i>	- The OpenWireFormat properties
<i>dataIn</i>	- stream to read marshaled form from
<i>bs</i>	- boolean stream to marshal to.

**Returns**

pointer to a new DataStructure Object

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.38 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString
( decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw (
    decaf::io::IOException ) [protected, virtual]
```

Performs Tight Unmarshaling of String Objects.

#### Parameters

<i>dataIn</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

#### Returns

the unmarshaled string.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.132.3.39 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes
( const std::vector< unsigned char > & data ) [static]
```

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

#### Parameters

<i>data</i>	- unsigned char data array pointer
-------------	------------------------------------

#### Returns

a string coded in hex that represents the data

```
6.132.3.40 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString
( const commands::TransactionId * txnId ) [static]
```

Converts the given transaction ID into a String.

#### Parameters

<i>txnId</i>	- TransactionId pointer
--------------	-------------------------

#### Returns

string representation of the id

6.132.3.41 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString  
( const commands::ProducerId * id ) [static]`

Converts the object to a String.

#### Parameters

<i>id</i>	- ProducerId pointer
-----------	----------------------

#### Returns

string representing the id

6.132.3.42 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString  
( const commands::MessageId * id ) [static]`

Converts the object to a String.

#### Parameters

<i>id</i>	- MessageId pointer
-----------	---------------------

#### Returns

string representing the id

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h`

## 6.133 activemq::commands::BaseDataStructure Class Reference

```
#include <src/main/activemq/commands/BaseDataStructure.h>
```

Inheritance diagram for `activemq::commands::BaseDataStructure`:

#### Public Member Functions

- virtual `~BaseDataStructure()`
- virtual bool `isMarshalAware()` const

*Determine if this object is aware of marshaling and should have its before and after marshaling methods called.*



- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed before an marshal.*
- virtual void **afterMarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed after an unmarshal.*
- virtual void **beforeUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed before an unmarshal.*
- virtual void **afterUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Perform any processing needed after an unmarshal.*
- virtual void **setMarshaledForm** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED, const std::vector< char > &data AMQCPP\_UNUSED)  
*Called to set the data to this object that will contain the objects marshaled form.*
- virtual std::vector< unsigned char > **getMarshaledForm** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED)  
*Called to get the data to this object that will contain the objects marshaled form.*
- virtual void **copyDataStructure** (const **DataStructure** \*src AMQCPP\_UNUSED)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value AMQCPP\_UNUSED) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*

### 6.133.1 Constructor & Destructor Documentation

- 6.133.1.1 virtual **activemq::commands::BaseDataStructure::~BaseDataStructure** ( )  
[inline, virtual]

### 6.133.2 Member Function Documentation

- 6.133.2.1 virtual void **activemq::commands::BaseDataStructure::afterMarshal** (**wireformat::WireFormat** \*wireFormat *AMQCPP\_UNUSED* ) throw ( decaf::io::IOException ) [inline, virtual]

Perform any processing needed after an unmarshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

6.133.2.2 `virtual void activemq::commands::BaseDataStructure::afterUnmarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Perform any processing needed after an unmarshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

Reimplemented in **activemq::commands::Message** (p. 2480), and **activemq::commands::WireFormatInfo** (p. 3914).

6.133.2.3 `virtual void activemq::commands::BaseDataStructure::beforeMarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Perform any processing needed before an marshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

Reimplemented in **activemq::commands::Message** (p. 2480), and **activemq::commands::WireFormatInfo** (p. 3915).

6.133.2.4 `virtual void activemq::commands::BaseDataStructure::beforeUnmarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

Perform any processing needed before an unmarshal.

#### Parameters

<i>wireformat</i>	- the OpenWireFormat object in use.
-------------------	-------------------------------------

6.133.2.5 `virtual void activemq::commands::BaseDataStructure::copyDataStructure ( const DataStructure *src AMQCPP_UNUSED ) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented in **activemq::commands::BooleanExpression** (p. 817).

6.133.2.6 `virtual bool activemq::commands::BaseDataStructure::equals ( const DataStructure *value AMQCPP_UNUSED ) const [inline, virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Referenced by `activemq::commands::BooleanExpression::equals()`, and `activemq::commands::BaseCommand::equals()`.

6.133.2.7 `virtual std::vector<unsigned char> activemq::commands::BaseDataStructure::getMarshaledForm ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) [inline, virtual]`

Called to get the data to this object that will contain the objects marshaled form.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

#### Returns

buffer that holds the objects data.

6.133.2.8 `virtual bool activemq::commands::BaseDataStructure::isMarshalAware ( ) const [inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

#### Returns

true if aware of marshaling

Implements **activemq::wireformat::MarshalAware** (p. 2446).

Reimplemented in **activemq::commands::ActiveMQMapMessage** (p. 339), **activemq::commands::Message** (p. 2486), and **activemq::commands::WireFormatInfo** (p. 3918).

6.133.2.9 `virtual void activemq::commands::BaseDataStructure::setMarshaledForm ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED ) [inline, virtual]`

Called to set the data to this object that will contain the objects marshaled form.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
<i>data</i>	- vector of object binary data

6.133.2.10 `virtual std::string activemq::commands::BaseDataStructure::toString ( ) const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p. 1632).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 177), **activemq::commands::ActiveMQDestination** (p. 214), **activemq::commands::ActiveMQDestination** (p. 302), **activemq::commands::ActiveMQMapMessage** (p. 344), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 416), **activemq::commands::ActiveMQQueue** (p. 457), **activemq::commands::ActiveMQStreamMessage** (p. 518), **activemq::commands::ActiveMQTempDestination** (p. 550), **activemq::commands::ActiveMQTempDestination** (p. 578), **activemq::commands::ActiveMQTempTopic** (p. 606), **activemq::commands::ActiveMQTextMessage** (p. 635), **activemq::commands::ActiveMQTopic** (p. 663), **activemq::commands::BaseCommand** (p. 729), **activemq::commands::BooleanExpression** (p. 817), **activemq::commands::BrokerId** (p. 831), **activemq::commands::BrokerInfo** (p. 861), **activemq::commands::Command** (p. 1169), **activemq::commands::ConnectionControl** (p. 1241), **activemq::commands::ConnectionError** (p. 1269), **activemq::commands::ConnectionId** (p. 1300), **activemq::commands::ConnectionInfo** (p. 1329), **activemq::commands::ConsumerControl** (p. 1372), **activemq::commands::ConsumerId** (p. 1401), **activemq::commands::ConsumerInfo** (p. 1432), **activemq::commands::ControlCommand** (p. 1462), **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse** (p. 1552), **activemq::commands::DestinationInfo** (p. 1695), **activemq::commands::DiscoveryEvent** (p. 1724), **activemq::commands::ExceptionResponse** (p. 1804), **activemq::commands::FlushCommand** (p. 1902), **activemq::commands::IntegerResponse** (p. 2056), **activemq::commands::JournalQueueAck** (p. 2119), **activemq::commands::JournalTopicAck** (p. 2147), **activemq::commands::JournalTrace** (p. 2174), **activemq::commands::JournalTransaction** (p. 2201), **activemq::commands::KeepAliveInfo** (p. 2228), **activemq::commands::LastPartialCommand** (p. 2262), **activemq::commands::LocalTransactionId** (p. 2310), **activemq::commands::Message** (p. 2490), **activemq::commands::MessageAck** (p. 2525), **activemq::commands::MessageDispatch** (p. 2558), **activemq::commands::MessageDispatchNot** (p. 2594), **activemq::commands::MessageId** (p. 2627), **activemq::commands::MessagePull** (p. 2699), **activemq::commands::NetworkBridgeFilter** (p. 2748), **activemq::commands::PartialCommand** (p. 2869), **activemq::commands::ProducerAck** (p. 2987), **activemq::commands::ProducerId** (p. 3018), **activemq::commands::ProducerInfo** (p. 3046), **activemq::commands::RemoveInfo**

(p. 3140), **activemq::commands::RemoveSubscriptionInfo** (p. 3168), **activemq::commands::ReplayCommand** (p. 3196), **activemq::commands::Response** (p. 3230), **activemq::commands::SessionId** (p. 3324), **activemq::commands::SessionInfo** (p. 3351), **activemq::commands::ShutdownInfo** (p. 3415), **activemq::commands::SubscriptionInfo** (p. 3619), **activemq::commands::TransactionId** (p. 3762), **activemq::commands::TransactionInfo** (p. 3788), **activemq::commands::WireFormatInfo** (p. 3922), and **activemq::commands::XATransactionId** (p. 3964).

Referenced by **activemq::commands::BooleanExpression::toString()**, and **activemq::commands::BaseCommand::toString()**.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseDataStructure.h`

## 6.134 binary\_function Class Reference

Inheritance diagram for `binary_function`:

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

## 6.135 decaf::net::BindException Class Reference

```
#include <src/main/decaf/net/BindException.h>
```

Inheritance diagram for `decaf::net::BindException`:

### Public Member Functions

- **BindException** () throw ()  
*Default Constructor.*
- **BindException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **BindException** (const **BindException** &ex) throw ()  
*Copy Constructor.*
- **BindException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BindException** (const std::exception \*cause) throw ()  
*Constructor.*

- **BindException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **BindException** \* **clone** () const  
*Clones this exception.*
- virtual ~**BindException** () throw ()

### 6.135.1 Constructor & Destructor Documentation

#### 6.135.1.1 decaf::net::BindException::BindException ( ) throw () [inline]

Default Constructor.

#### 6.135.1.2 decaf::net::BindException::BindException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

##### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

#### 6.135.1.3 decaf::net::BindException::BindException ( const BindException & ex ) throw () [inline]

Copy Constructor.

##### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

#### 6.135.1.4 decaf::net::BindException::BindException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

##### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.135.1.5 `decaf::net::BindException::BindException ( const std::exception * cause ) throw ()`  
`[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.135.1.6 `decaf::net::BindException::BindException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.135.1.7 `virtual decaf::net::BindException::~~BindException ( ) throw ()` `[inline, virtual]`

## 6.135.2 Member Function Documentation

6.135.2.1 `virtual BindException* decaf::net::BindException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3467).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BindException.h`

## 6.136 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

```
#include <src/main/decaf/io/BlockingByteArrayInputStream.h>
```

Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

### Public Member Functions

- **BlockingByteArrayInputStream** ()  
*Default Constructor - uses a default internal buffer.*
- **BlockingByteArrayInputStream** (const unsigned char \*buffer, int bufferSize)  
*Constructor that initializes the internal buffer.*
- virtual ~**BlockingByteArrayInputStream** ()
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize)

- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.*  
*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*  
*The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

#### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2002) freeing any resources that might have been aquired during the lifetime of this stream.*  
*The default implementation of this method does nothing.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedC )

*Skips over and discards n bytes of data from this input stream.*  
*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*  
*The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

#### Parameters

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------



**Returns***total bytes skipped***Exceptions**

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
UnsupportedOperationException	<i>if the concrete stream class does not support skipping bytes.</i>

**Protected Member Functions**

- virtual int **doReadByte** () throw ( IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**6.136.1 Detailed Description**

This is a blocking version of a byte buffer stream.

Read operations block until the requested data becomes available in the internal buffer via a call to `setByteArray`.

**6.136.2 Constructor & Destructor Documentation****6.136.2.1 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ( )**

Default Constructor - uses a default internal buffer.

**6.136.2.2 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ( const unsigned char \* *buffer*, int *bufferSize* )**

Constructor that initializes the internal buffer.

**See also**

**setByteArray** (p. 803).

**6.136.2.3 virtual decaf::io::BlockingByteArrayInputStream::~~BlockingByteArrayInputStream ( ) [virtual]****6.136.3 Member Function Documentation****6.136.3.1 virtual int decaf::io::BlockingByteArrayInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]**

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

### Returns

the number of bytes available on this input stream.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2004).

6.136.3.2 `virtual void decaf::io::BlockingByteArrayInputStream::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2004).

6.136.3.3 `virtual int decaf::io::BlockingByteArrayInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented from **decaf::io::InputStream** (p. 2005).

6.136.3.4 `virtual int decaf::io::BlockingByteArrayInputStream::doReadByte ( ) throw ( IOException ) [protected, virtual]`

Implements **decaf::io::InputStream** (p. 2005).

6.136.3.5 `virtual void decaf::io::BlockingByteArrayInputStream::setByteArray ( const unsigned char * buffer, int bufferSize ) [virtual]`

**6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference**

6.136.3.6 virtual long long decaf::io::BlockingByteArrayInputStream::skip  
( long long num ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p.2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters**

num	The number of bytes to skip.
-----	------------------------------

**Returns**

total bytes skipped

**Exceptions**

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p.2010).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BlockingByteArrayInputStream.h**

**6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference**

A **decaf::util::Queue** (p.3094) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

#include <src/main/decaf/util/concurrent/BlockingQueue.h>

Inheritance diagram for decaf::util::concurrent::BlockingQueue< E >:

## Public Member Functions

- virtual **~BlockingQueue** ()
- virtual void **put** (const E &value)=0 throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, waiting if necessary for space to become available.*
- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.*
- virtual E **take** ()=0 throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.*
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.*
- virtual int **remainingCapacity** () const =0  
*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or Integer::MAX\_VALUE if there is no intrinsic limit.*
- virtual std::size\_t **drainTo** (**Collection**< E > &c)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes all available elements from this queue and adds them to the given collection.*
- virtual std::size\_t **drainTo** (**Collection**< E > &c, std::size\_t maxElements)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Removes at most the given number of available elements from this queue and adds them to the given collection.*

### 6.137.1 Detailed Description

template<typename E>class decaf::util::concurrent::BlockingQueue< E >

A **decaf::util::Queue** (p. 3094) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

**BlockingQueue** (p. 804) methods come in four forms, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: one throws an exception, the second returns a special value (either `true` or `false`, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:

## 6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference 607

	<i>Throws exception</i>	<i>Boolean Flag</i>	<i>Blocks</i>	<i>Times out</i>
<b>Insert</b>	<b>add(e)</b> (p. 165)	<b>offer(e)</b> (p. 808)	<b>put(e)</b> (p. 809)	<b>offer(e, time, unit)</b> (p. ??)
<b>Remove</b>	<b>remove()</b> (p. 167)	<b>poll()</b> (p. 809)	<b>take()</b> (p. 810)	<b>poll(time, unit)</b> (p. ??)
<b>Examine</b>	<b>element()</b> (p. 166)	<b>peek()</b> (p. 3097)	<i>not applicable</i>	<i>not applicable</i>

A **BlockingQueue** (p. 804) may be capacity bounded. At any given time it may have a `remainingCapacity` beyond which no additional elements can be put without blocking. A **BlockingQueue** (p. 804) without any intrinsic capacity constraints always reports a remaining capacity of `Integer::MAX_VALUE`.

**BlockingQueue** (p. 804) implementations are designed to be used primarily for producer-consumer queues, but additionally support **decaf::util::Collection** (p. 1155) interface. So, for example, it is possible to remove an arbitrary element from a queue using `remove(x)`. However, such operations are in general *not* performed very efficiently, and are intended for only occasional use, such as when a queued message is cancelled.

**BlockingQueue** (p. 804) implementations are thread-safe. All queuing methods achieve their effects atomically using internal locks or other forms of concurrency control. However, the *bulk* **Collection** (p. 1155) operations `addAll`, `containsAll`, `retainAll` and `removeAll` are *not* necessarily performed atomically unless specified otherwise in an implementation. So it is possible, for example, for `addAll(c)` to fail (throwing an exception) after adding only some of the elements in `c`.

A **BlockingQueue** (p. 804) does *not* intrinsically support any kind of "close" or "shutdown" operation to indicate that no more items will be added. The needs and usage of such features tend to be implementation-dependent. For example, a common tactic is for producers to insert special *end-of-stream* or *poison* objects, that are interpreted accordingly when taken by consumers.

Usage example, based on a typical producer-consumer scenario. Note that a **BlockingQueue** (p. 804) can safely be used with multiple producers and multiple consumers.

```
class Producer : public Runnable {
private:

    BlockingQueue* queue;

public:

    Producer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { queue->put( produce() ); }
        } catch( InterruptedException& ex ) { ... handle ... }
    }

    Object produce() { ... }
}
```

```

class Consumer : public Runnable {
private:

    BlockingQueue* queue;

public:

    Consumer( BlockingQueue* q ) : queue( q ) {}

    virtual void run() {
        try {
            while( true ) { consume( queue->take() (p.810) ); }
        } catch( InterruptedException& ex ) { ... handle ...}
    }

    void consume( Object& x ) { ... }
}

int main( int argc, char** argv ) {

    BlockingQueue (p.804) q = new SomeQueueImplementation();
    Producer p( &q );
    Consumer c1( &q );
    Consumer c2( &q );
    Thread t1( &p ).start();
    Thread t2( &c1 ).start();
    Thread t3( &c2 ).start();
}

```

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a **BlockingQueue** (p.804) *happen-before* actions subsequent to the access or removal of that element from the **BlockingQueue** (p.804) in another thread.

**Since**

1.0

## 6.137.2 Constructor & Destructor Documentation

6.137.2.1 `template<typename E> virtual decaf::util::concurrent::BlockingQueue< E >::~~BlockingQueue( ) [inline, virtual]`

## 6.137.3 Member Function Documentation

## 6.137 `decaf::util::concurrent::BlockingQueue< E >` Class Template Reference 309

```
6.137.3.1 template<typename E> virtual std::size_t
    decaf::util::concurrent::BlockingQueue< E >::drainTo ( Collection< E >
    & c ) throw ( decaf::lang::exceptions::UnsupportedOperationException,
    decaf::lang::exceptions::IllegalArgumentException ) [pure
    virtual]
```

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

### Parameters

<code>c</code>	the collection to transfer elements into
----------------	--

### Returns

the number of elements transferred

### Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3663).

```
6.137.3.2 template<typename E> virtual std::size_t
    decaf::util::concurrent::BlockingQueue< E >::drainTo
    ( Collection< E > & c, std::size_t maxElements ) throw (
    decaf::lang::exceptions::UnsupportedOperationException,
    decaf::lang::exceptions::IllegalArgumentException ) [pure
    virtual]
```

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection `c` may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

### Parameters

<code>c</code>	the collection to transfer elements into
----------------	--

<i>maxElements</i>	the maximum number of elements to transfer
--------------------	--

**Returns**

the number of elements transferred

**Exceptions**

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3664).

```
6.137.3.3  template<typename E > virtual bool
            decaf::util::concurrent::BlockingQueue< E >::offer
            ( const E & e, long timeout, const TimeUnit & unit )
            throw ( decaf::lang::exceptions::InterruptedException,
                    decaf::lang::exceptions::NullPointerException,
                    decaf::lang::exceptions::IllegalArgumentException ) [pure
            virtual]
```

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

**Parameters**

<i>e</i>	the element to add
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a <b>TimeUnit</b> (p. 3748) determining how to interpret the <i>timeout</i> parameter

**Returns**

`true` if successful, or `false` if the specified waiting time elapses before space is available

**Exceptions**

<i>InterruptedException</i>	if interrupted while waiting
<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3666).



## 6.137 decaf::util::concurrent::BlockingQueue< E > Class Template Reference 611

6.137.3.4 `template<typename E > virtual bool  
decaf::util::concurrent::BlockingQueue< E >::poll ( E  
& result, long long timeout, const TimeUnit & unit ) throw (   
decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

### Parameters

<i>result</i>	the referenced value that will be assigned the value retrieved from the <b>Queue</b> (p. 3094). Undefined if this methods returned false.
<i>timeout</i>	how long to wait before giving up, in units of <i>unit</i>
<i>unit</i>	a <b>TimeUnit</b> (p. 3748) determining how to interpret the <i>timeout</i> parameter.

### Returns

`true` if successful or `false` if the specified waiting time elapses before an element is available.

### Exceptions

<i>InterruptedException</i>	if interrupted while waiting
-----------------------------	------------------------------

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3667).

6.137.3.5 `template<typename E > virtual void  
decaf::util::concurrent::BlockingQueue< E >::put (   
const E & value ) throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException ) [pure  
virtual]`

Inserts the specified element into this queue, waiting if necessary for space to become available.

### Parameters

<i>value</i>	the element to add
--------------	--------------------

### Exceptions

<i>InterruptedException</i>	if interrupted while waiting
<i>NullPointerException</i>	if the specified element is null
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3668).

6.137.3.6 `template<typename E> virtual int decaf::util::concurrent::BlockingQueue<E>::remainingCapacity ( ) const [pure virtual]`

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

#### Returns

the remaining capacity

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3668).

6.137.3.7 `template<typename E> virtual E decaf::util::concurrent::BlockingQueue<E>::take ( ) throw ( decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

#### Returns

the head of this queue

#### Exceptions

<code>InterruptedException</code>	if interrupted while waiting
-----------------------------------	------------------------------

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p. 3669).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BlockingQueue.h`

## 6.138 decaf::lang::Boolean Class Reference

```
#include <src/main/decaf/lang/Boolean.h>
```

Inheritance diagram for `decaf::lang::Boolean`:

#### Public Member Functions

- **Boolean** (bool value)

- **Boolean** (const std::string &value)
- virtual  $\sim$ **Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const  
*Compares this **Boolean** (p. 810) instance with another.*
- virtual bool **operator==** (const **Boolean** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Boolean** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const  
*Compares this **Boolean** (p. 810) instance with another.*
- virtual bool **operator==** (const bool &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const bool &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const bool &b) const

### Static Public Member Functions

- static **Boolean valueOf** (bool value)
- static **Boolean valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)  
*Parses the **String** (p. 3610) passed and extracts an bool.*
- static std::string **toString** (bool value)  
*Converts the bool to a **String** (p. 3610) representation.*

### Static Public Attributes

- static const **Boolean \_FALSE**  
*The Class object representing the primitive false boolean.*
- static const **Boolean \_TRUE**  
*The Class object representing the primitive type boolean.*

## 6.138.1 Constructor & Destructor Documentation

### 6.138.1.1 decaf::lang::Boolean::Boolean ( bool value )

#### Parameters

<i>value</i>	- primitive boolean to wrap.
--------------	------------------------------

6.138.1.2 `decaf::lang::Boolean::Boolean ( const std::string & value )`

#### Parameters

<i>value</i>	- <b>String</b> (p. 3610) value to convert to a boolean.
--------------	--

6.138.1.3 `virtual decaf::lang::Boolean::~~Boolean ( ) [inline, virtual]`

### 6.138.2 Member Function Documentation

6.138.2.1 `bool decaf::lang::Boolean::booleanValue ( ) const [inline]`

#### Returns

the primitive boolean value of this object

6.138.2.2 `virtual int decaf::lang::Boolean::compareTo ( const Boolean & b ) const [virtual]`

Compares this **Boolean** (p. 810) instance with another.

#### Parameters

<i>b</i>	- the <b>Boolean</b> (p. 810) instance to be compared
----------	---

#### Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements `decaf::lang::Comparable< Boolean >` (p. 1187).

6.138.2.3 `virtual int decaf::lang::Boolean::compareTo ( const bool & b ) const [virtual]`

Compares this **Boolean** (p. 810) instance with another.

#### Parameters

<i>b</i>	- the <b>Boolean</b> (p. 810) instance to be compared
----------	---

#### Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements `decaf::lang::Comparable< bool >` (p. 1187).

6.138.2.4 `bool decaf::lang::Boolean::equals ( const bool & b ) const` `[inline, virtual]`

#### Returns

true if the two **Boolean** (p. 810) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 1188).

6.138.2.5 `bool decaf::lang::Boolean::equals ( const Boolean & b ) const` `[inline, virtual]`

#### Returns

true if the two **Boolean** (p. 810) Objects have the same value.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1188).

6.138.2.6 `virtual bool decaf::lang::Boolean::operator< ( const bool & value ) const` `[virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1188).

6.138.2.7 `virtual bool decaf::lang::Boolean::operator< ( const Boolean & value ) const` `[virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1188).

6.138.2.8 `virtual bool decaf::lang::Boolean::operator== ( const bool & value ) const`  
[virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1189).

6.138.2.9 `virtual bool decaf::lang::Boolean::operator== ( const Boolean & value ) const`  
[virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Boolean** > (p. 1189).

6.138.2.10 `static bool decaf::lang::Boolean::parseBoolean ( const std::string & value )`  
[static]

Parses the **String** (p. 3610) passed and extracts an bool.

#### Parameters

<i>value</i>	The std::string value to parse
--------------	--------------------------------

#### Returns

bool value

6.138.2.11 `static std::string decaf::lang::Boolean::toString ( bool value )` [static]

Converts the bool to a **String** (p. 3610) representation.

**Parameters**

<i>value</i>	The bool value to convert.
--------------	----------------------------

**Returns**

std::string representation of the bool value passed.

**6.138.2.12 std::string decaf::lang::Boolean::toString ( ) const****Returns**

the string representation of this Booleans value.

**6.138.2.13 static Boolean decaf::lang::Boolean::valueOf ( bool value ) [static]****Parameters**

<i>value</i>	The bool value to convert to a <b>Boolean</b> (p. 810) instance.
--------------	--

**Returns**

a **Boolean** (p. 810) instance of the primitive boolean value

**6.138.2.14 static Boolean decaf::lang::Boolean::valueOf ( const std::string & value ) [static]****Parameters**

<i>value</i>	The std::string value to convert to a <b>Boolean</b> (p. 810) instance.
--------------	---

**Returns**

a **Boolean** (p. 810) instance of the string value

**6.138.3 Field Documentation****6.138.3.1 const Boolean decaf::lang::Boolean::\_FALSE [static]**

The Class object representing the primitive false boolean.

**6.138.3.2 const Boolean decaf::lang::Boolean::\_TRUE [static]**

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Boolean.h**

## 6.139 activemq::commands::BooleanExpression Class Reference

```
#include <src/main/activemq/commands/BooleanExpression.h>
```

Inheritance diagram for `activemq::commands::BooleanExpression`:

### Public Member Functions

- **BooleanExpression** ()
- virtual **~BooleanExpression** ()
- virtual **DataStructure \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src AMQCPP\_UNUSED)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*

### 6.139.1 Constructor & Destructor Documentation

6.139.1.1 `activemq::commands::BooleanExpression::BooleanExpression ( )` [inline]

6.139.1.2 `virtual activemq::commands::BooleanExpression::~~BooleanExpression ( )`  
 [inline, virtual]

### 6.139.2 Member Function Documentation

6.139.2.1 `virtual DataStructure* activemq::commands::BooleanExpression::cloneDataStructure ( ) const` [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1628).



6.139.2.2 `virtual void activemq::commands::BooleanExpression::copyDataStructure ( const DataStructure *src AMQCPP_UNUSED ) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object

Reimplemented from **activemq::commands::BaseDataStructure** (p. 795).

6.139.2.3 `virtual bool activemq::commands::BooleanExpression::equals ( const DataStructure * value ) const [inline, virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

References **activemq::commands::BaseDataStructure::equals()**.

6.139.2.4 `virtual std::string activemq::commands::BooleanExpression::toString ( ) const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

References **activemq::commands::BaseDataStructure::toString()**.

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BooleanExpression.h**

## 6.140 activemq::wireformat::openwire::utils::BooleanStream Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

```
#include <src/main/activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Public Member Functions

- **BooleanStream** ()
- virtual **~BooleanStream** ()
- bool **readBoolean** () throw ( `decaf::io::IOException` )  
*Read a boolean data element from the internal data buffer.*
- void **writeBoolean** (bool value) throw ( `decaf::io::IOException` )  
*Writes a Boolean value to the internal data buffer.*
- void **marshal** (`decaf::io::DataOutputStream` \*dataOut) throw ( `decaf::io::IOException` )  
*Marshal the data to a DataOutputStream.*
- void **marshal** (std::vector< unsigned char > &dataOut)  
*Marshal the data to a STL vector of unsigned chars.*
- void **unmarshal** (`decaf::io::DataInputStream` \*dataIn) throw ( `decaf::io::IOException` )  
*Unmarshal a Boolean data stream from the Input Stream.*
- void **clear** ()  
*Clears to old position markers, data starts at the beginning.*
- int **marshalledSize** ()  
*Calc the size that data is marshalled to.*

#### 6.140.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.

## 6.140 activemq::wireformat::openwire::utils::BooleanStream Class Reference821

### 6.140.2 Constructor & Destructor Documentation

6.140.2.1 `activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ( )`

6.140.2.2 `virtual activemq::wireformat::openwire::utils::BooleanStream::~~BooleanStream ( )`  
[inline, virtual]

### 6.140.3 Member Function Documentation

6.140.3.1 `void activemq::wireformat::openwire::utils::BooleanStream::clear ( )`

Clears to old position markers, data starts at the beginning.

6.140.3.2 `void activemq::wireformat::openwire::utils::BooleanStream::marshal ( std::vector< unsigned char > & dataOut )`

Marshal the data to a STL vector of unsigned chars.

#### Parameters

<i>dataOut</i>	- reference to a vector to write the data to.
----------------	---

6.140.3.3 `void activemq::wireformat::openwire::utils::BooleanStream::marshal ( decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )`

Marshal the data to a DataOutputStream.

#### Parameters

<i>dataOut</i>	- Stream to write the data to.
----------------	--------------------------------

6.140.3.4 `int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize ( )`

Calc the size that data is marshalled to.

#### Returns

int size of marshalled data.

6.140.3.5 `bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean ( ) throw ( decaf::io::IOException )`

Read a boolean data element from the internal data buffer.

#### Returns

boolean from the stream

6.140.3.6 `void activemq::wireformat::openwire::utils::BooleanStream::unmarshal ( decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`

Unmarshal a Boolean data stream from the Input Stream.

#### Parameters

<i>dataIn</i>	- Input Stream to read data from.
---------------	-----------------------------------

6.140.3.7 `void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean ( bool value ) throw ( decaf::io::IOException )`

Writes a Boolean value to the internal data buffer.

#### Parameters

<i>value</i>	- boolean data to write.
--------------	--------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/BooleanStream.h`

## 6.141 decaf::util::concurrent::BrokenBarrierException Class Reference

```
#include <src/main/decaf/util/concurrent/BrokenBarrierException.h>
```

Inheritance diagram for `decaf::util::concurrent::BrokenBarrierException`:

### Public Member Functions

- **BrokenBarrierException** () throw ()  
*Default Constructor.*
- **BrokenBarrierException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **BrokenBarrierException** (const **BrokenBarrierException** &ex) throw ()  
*Copy Constructor.*
- **BrokenBarrierException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **BrokenBarrierException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- **BrokenBarrierException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **BrokenBarrierException** \* clone () const  
*Clones this exception.*
- virtual ~**BrokenBarrierException** () throw ()

### 6.141.1 Constructor & Destructor Documentation

6.141.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( ) throw ()  
 [inline]

Default Constructor.

6.141.1.2 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const decaf::lang::Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

ex	An exception that should become this type of Exception
----	--

6.141.1.3 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const BrokenBarrierException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

ex	The Exception to copy in this new instance.
----	---

6.141.1.4 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const std::exception \* cause ) throw () [inline]

Constructor.

#### Parameters

cause	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
-------	--

6.141.1.5 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.141.1.6 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.141.1.7 `virtual decaf::util::concurrent::BrokenBarrierException::~BrokenBarrierException ( ) throw () [inline, virtual]`

## 6.141.2 Member Function Documentation

6.141.2.1 `virtual BrokenBarrierException* decaf::util::concurrent::BrokenBarrierException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**BrokenBarrierException.h**

## 6.142 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

```
#include <src/main/activemq/commands/BrokerError.h>
```

Inheritance diagram for activemq::commands::BrokerError:

### Data Structures

- struct **StackTraceElement**

### Public Member Functions

- **BrokerError** ()
- virtual **~BrokerError** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.*
- virtual **BrokerError** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- virtual const std::string & **getMessage** () const  
*Gets the string holding the error message.*
- virtual void **setMessage** (const std::string &message)  
*Sets the string that contains the error **Message** (p. 2475).*
- virtual const std::string & **getExceptionClass** () const  
*Gets the string holding the Exception Class name.*
- virtual void **setExceptionClass** (const std::string &exceptionClass)  
*Sets the string that contains the Exception Class name.*
- virtual const **decaf::lang::Pointer**< **BrokerError** > & **getCause** () const

*Gets the Broker Error that caused this exception.*

- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)

*Sets the Broker Error that caused this exception.*

- virtual const std::vector< **decaf::lang::Pointer**< **StackTraceElement** > > &**getStackTraceElements** () const

*Gets the Stack Trace Elements for the Exception.*

- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer**< **StackTraceElement** > > &stackTraceElements)

*Sets the Stack Trace Elements for this Exception.*

### 6.142.1 Detailed Description

This class represents an Exception sent from the Broker.

The Broker sends java Throwables, so we must mimic its structure here.

### 6.142.2 Constructor & Destructor Documentation

6.142.2.1 **activemq::commands::BrokerError::BrokerError** ( )

6.142.2.2 **virtual activemq::commands::BrokerError::~~BrokerError** ( ) [virtual]

### 6.142.3 Member Function Documentation

6.142.3.1 **virtual BrokerError\*** **activemq::commands::BrokerError::cloneDataStructure** ( )  
const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

References **copyDataStructure()**.

6.142.3.2 **virtual void** **activemq::commands::BrokerError::copyDataStructure** ( const **DataStructure** \* **src** ) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

#### Returns

src - Source Object



Reimplemented from **activemq::commands::BaseCommand** (p. 724).

Referenced by cloneDataStructure().

```
6.142.3.3 virtual const decaf::lang::Pointer<BrokerError>&
activemq::commands::BrokerError::getCause ( ) const [inline,
virtual]
```

Gets the Broker Error that caused this exception.

#### Returns

Broker Error Pointer

```
6.142.3.4 virtual unsigned char activemq::commands::BrokerError::getDataStructureType ( )
const [inline, virtual]
```

Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.

#### Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1631).

```
6.142.3.5 virtual const std::string& activemq::commands::BrokerError::getExceptionClass ( )
const [inline, virtual]
```

Gets the string holding the Exception Class name.

#### Returns

Exception Class name

```
6.142.3.6 virtual const std::string& activemq::commands::BrokerError::getMessage ( ) const
[inline, virtual]
```

Gets the string holding the error message.

#### Returns

String **Message** (p. 2475)

6.142.3.7 `virtual const std::vector< decaf::lang::Pointer< StackTraceElement > > & activemq::commands::BrokerError::getStackTraceElements ( ) const [inline, virtual]`

Gets the Stack Trace Elements for the Exception.

#### Returns

Stack Trace Elements

6.142.3.8 `virtual void activemq::commands::BrokerError::setCause ( const decaf::lang::Pointer< BrokerError > & cause ) [inline, virtual]`

Sets the Broker Error that caused this exception.

#### Parameters

<i>cause</i>	- Broker Error
--------------	----------------

6.142.3.9 `virtual void activemq::commands::BrokerError::setExceptionClass ( const std::string & exceptionClass ) [inline, virtual]`

Sets the string that contains the Exception Class name.

#### Parameters

<i>exception-Class</i>	- String Exception Class name
------------------------	-------------------------------

6.142.3.10 `virtual void activemq::commands::BrokerError::setMessage ( const std::string & message ) [inline, virtual]`

Sets the string that contains the error **Message** (p. 2475).

#### Parameters

<i>message</i>	- String Error <b>Message</b> (p. 2475)
----------------	---

6.142.3.11 `virtual void activemq::commands::BrokerError::setStackTraceElements ( const std::vector< decaf::lang::Pointer< StackTraceElement > > & stackTraceElements ) [inline, virtual]`

Sets the Stack Trace Elements for this Exception.

**Parameters**

<i>stack-TraceElements</i>	- Stack Trace Elements
----------------------------	------------------------

6.142.3.12 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::commands::BrokerError::visit ( activemq::state::CommandVisitor  
 * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

**6.143 activemq::exceptions::BrokerException Class Reference**

```
#include <src/main/activemq/exceptions/BrokerException.h>
```

Inheritance diagram for `activemq::exceptions::BrokerException`:

**Public Member Functions**

- **BrokerException** () throw ()
- **BrokerException** (const **exceptions::ActiveMQException** &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()
- **BrokerException** (const char \*file, const int lineNumber, const **commands::BrokerError** \*error) throw ()
- virtual **BrokerException** \* **clone** () const  
*Clones this exception.*
- virtual ~**BrokerException** () throw ()

### 6.143.1 Constructor & Destructor Documentation

- 6.143.1.1 `activemq::exceptions::BrokerException::BrokerException ( ) throw ()` `[inline]`
- 6.143.1.2 `activemq::exceptions::BrokerException::BrokerException ( const exceptions::ActiveMQException & ex ) throw ()` `[inline]`
- 6.143.1.3 `activemq::exceptions::BrokerException::BrokerException ( const BrokerException & ex ) throw ()` `[inline]`
- 6.143.1.4 `activemq::exceptions::BrokerException::BrokerException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`
- 6.143.1.5 `activemq::exceptions::BrokerException::BrokerException ( const char * file, const int lineNumber, const commands::BrokerError * error ) throw ()` `[inline]`

References `decaf::lang::Exception::getMessage()`.

- 6.143.1.6 `virtual activemq::exceptions::BrokerException::~~BrokerException ( ) throw ()`  
`[inline, virtual]`

### 6.143.2 Member Function Documentation

- 6.143.2.1 `virtual BrokerException* activemq::exceptions::BrokerException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `activemq::exceptions::ActiveMQException` (p. 330).

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/BrokerException.h`

## 6.144 `activemq::commands::BrokerId` Class Reference

```
#include <src/main/activemq/commands/BrokerId.h>
```

Inheritance diagram for `activemq::commands::BrokerId`:

### Public Types

- `typedef decaf::lang::PointerComparator< BrokerId > COMPARATOR`

## Public Member Functions

- **BrokerId** ()
- **BrokerId** (const **BrokerId** &other)
- virtual ~**BrokerId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **BrokerId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **BrokerId** &value) const
- virtual bool **equals** (const **BrokerId** &value) const
- virtual bool **operator==** (const **BrokerId** &value) const
- virtual bool **operator<** (const **BrokerId** &value) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

## Static Public Attributes

- static const unsigned char **ID\_BROKERID** = 124

## Protected Attributes

- std::string **value**

### 6.144.1 Member Typedef Documentation

- 6.144.1.1 typedef decaf::lang::PointerComparator<BrokerId>  
activemq::commands::BrokerId::COMPARATOR

### 6.144.2 Constructor & Destructor Documentation

- 6.144.2.1 activemq::commands::BrokerId::BrokerId ( )

6.144.2.2 `activemq::commands::BrokerId::BrokerId ( const BrokerId & other )`

6.144.2.3 `virtual activemq::commands::BrokerId::~~BrokerId ( ) [virtual]`

### 6.144.3 Member Function Documentation

6.144.3.1 `virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1628).

6.144.3.2 `virtual int activemq::commands::BrokerId::compareTo ( const BrokerId & value ) const [virtual]`

6.144.3.3 `virtual void activemq::commands::BrokerId::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Implements `activemq::commands::DataStructure` (p. 1629).

6.144.3.4 `virtual bool activemq::commands::BrokerId::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1630).

6.144.3.5 `virtual bool activemq::commands::BrokerId::equals ( const BrokerId & value ) const`  
[virtual]

6.144.3.6 `virtual unsigned char activemq::commands::BrokerId::getDataStructureType ( )`  
`const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.144.3.7 `virtual const std::string& activemq::commands::BrokerId::getValue ( ) const`  
[virtual]

6.144.3.8 `virtual std::string& activemq::commands::BrokerId::getValue ( )` [virtual]

6.144.3.9 `virtual bool activemq::commands::BrokerId::operator< ( const BrokerId & value )`  
`const` [virtual]

6.144.3.10 `BrokerId& activemq::commands::BrokerId::operator= ( const BrokerId & other )`

6.144.3.11 `virtual bool activemq::commands::BrokerId::operator== ( const BrokerId & value )`  
`const` [virtual]

6.144.3.12 `virtual void activemq::commands::BrokerId::setValue ( const std::string & value )`  
[virtual]

6.144.3.13 `virtual std::string activemq::commands::BrokerId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

### 6.144.4 Field Documentation

6.144.4.1 `const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124`  
[static]

6.144.4.2 `std::string activemq::commands::BrokerId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerId.h`

## 6.145 `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 832).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshal
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller`:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*



### 6.145.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 832).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.145.2 Constructor & Destructor Documentation

6.145.2.1 `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::BrokerIdMarshaller ( ) [inline]`

6.145.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::~~BrokerIdMarshaller ( ) [inline, virtual]`

### 6.145.3 Member Function Documentation

6.145.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.145.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.145.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.145.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.145.3.5 virtual int activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.145.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.145.3.7 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h`

## 6.146 `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 836).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller`:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.146.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 836).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.146.2 Constructor & Destructor Documentation

6.146.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::BrokerIdMarshaller ( ) [inline]`

6.146.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::~~BrokerIdMarshaller ( ) [inline, virtual]`

### 6.146.3 Member Function Documentation

6.146.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.146.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.146.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.146.3.4 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.146.3.5 virtual int activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.146.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.146.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerIdMarshaller.h**

## 6.147 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 840).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*



### 6.147.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 840).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.147.2 Constructor & Destructor Documentation

6.147.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::BrokerIdMarshaller ( ) [inline]`

6.147.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::~~BrokerIdMarshaller ( ) [inline, virtual]`

### 6.147.3 Member Function Documentation

6.147.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.147.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.147.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.147.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.147.3.5 virtual int activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## 6.147 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference 845

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.147.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.147.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerIdMarshaller.h**

## 6.148 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 844).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshal
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.148.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 844).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.148.2 Constructor & Destructor Documentation

6.148.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::BrokerIdMarshaller ( ) [inline]`

6.148.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::~~BrokerIdMarshaller ( ) [inline, virtual]`

### 6.148.3 Member Function Documentation

6.148.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.148.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.148.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.148.3.4 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.148.3.5 virtual int activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.148.3.6 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.148.3.7 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerIdMarshaller.h**

## 6.149 activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 848).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshal
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*



### 6.149.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 848).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.149.2 Constructor & Destructor Documentation

6.149.2.1 `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::BrokerIdMarshaller ( ) [inline]`

6.149.2.2 `virtual activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::~~BrokerIdMarshaller ( ) [inline, virtual]`

### 6.149.3 Member Function Documentation

6.149.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.149.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.149.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.149.3.4 virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.149.3.5 virtual int activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.149.3.6 virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.149.3.7 virtual void activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**BrokerIdMarshaller.h**

## 6.150 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 852).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller:

### Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.150.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 852).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.150.2 Constructor & Destructor Documentation

6.150.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::BrokerIdMarshaller ( ) [inline]`

6.150.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::~~BrokerIdMarshaller ( ) [inline, virtual]`

### 6.150.3 Member Function Documentation

6.150.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.150.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.150.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.150.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.150.3.5 virtual int activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.150.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.150.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshall/v2/**BrokerIdMarshaller.h**

## 6.151 activemq::commands::BrokerInfo Class Reference

```
#include <src/main/activemq/commands/BrokerInfo.h>
```

Inheritance diagram for activemq::commands::BrokerInfo:

### Public Member Functions

- **BrokerInfo** ()
- virtual **~BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **BrokerInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &brokerId)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &brokerURL)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &peerBrokerInfos)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool slaveBroker)



- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool **masterBroker**)
- virtual bool **isFaultTolerantConfiguration** () const
- virtual void **setFaultTolerantConfiguration** (bool **faultTolerantConfiguration**)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool **duplexConnection**)
- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool **networkConnection**)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long **connectionId**)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &**brokerUploadUrl**)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &**networkProperties**)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_BROKERINFO** = 2

### Protected Attributes

- **Pointer< BrokerId > brokerId**
- std::string **brokerURL**
- std::vector< decaf::lang::Pointer< BrokerInfo > > **peerBrokerInfos**
- std::string **brokerName**
- bool **slaveBroker**
- bool **masterBroker**
- bool **faultTolerantConfiguration**
- bool **duplexConnection**
- bool **networkConnection**
- long long **connectionId**
- std::string **brokerUploadUrl**
- std::string **networkProperties**

### 6.151.1 Constructor & Destructor Documentation

6.151.1.1 `activemq::commands::BrokerInfo::BrokerInfo ( )`

6.151.1.2 `virtual activemq::commands::BrokerInfo::~~BrokerInfo ( ) [virtual]`

### 6.151.2 Member Function Documentation

6.151.2.1 `virtual BrokerInfo* activemq::commands::BrokerInfo::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.151.2.2 `virtual void activemq::commands::BrokerInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.151.2.3 `virtual bool activemq::commands::BrokerInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

- 6.151.2.4 `virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ( ) const [virtual]`
- 6.151.2.5 `virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ( ) [virtual]`
- 6.151.2.6 `virtual std::string& activemq::commands::BrokerInfo::getBrokerName ( ) [virtual]`
- 6.151.2.7 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerName ( ) const [virtual]`
- 6.151.2.8 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ( ) const [virtual]`
- 6.151.2.9 `virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ( ) [virtual]`
- 6.151.2.10 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL ( ) const [virtual]`
- 6.151.2.11 `virtual std::string& activemq::commands::BrokerInfo::getBrokerURL ( ) [virtual]`
- 6.151.2.12 `virtual long long activemq::commands::BrokerInfo::getConnectionId ( ) const [virtual]`
- 6.151.2.13 `virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

- 6.151.2.14 `virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties ( ) const [virtual]`
- 6.151.2.15 `virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties ( ) [virtual]`
- 6.151.2.16 `virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ( ) const [virtual]`

- 6.151.2.17 `virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos ( ) [virtual]`
- 6.151.2.18 `virtual bool activemq::commands::BrokerInfo::isBrokerInfo ( ) const [inline, virtual]`

### Returns

an answer of true to the **isBrokerInfo()** (p. 860) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 726).

- 6.151.2.19 `virtual bool activemq::commands::BrokerInfo::isDuplexConnection ( ) const [virtual]`
- 6.151.2.20 `virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration ( ) const [virtual]`
- 6.151.2.21 `virtual bool activemq::commands::BrokerInfo::isMasterBroker ( ) const [virtual]`
- 6.151.2.22 `virtual bool activemq::commands::BrokerInfo::isNetworkConnection ( ) const [virtual]`
- 6.151.2.23 `virtual bool activemq::commands::BrokerInfo::isSlaveBroker ( ) const [virtual]`
- 6.151.2.24 `virtual void activemq::commands::BrokerInfo::setBrokerId ( const Pointer< BrokerId > & brokerId ) [virtual]`
- 6.151.2.25 `virtual void activemq::commands::BrokerInfo::setBrokerName ( const std::string & brokerName ) [virtual]`
- 6.151.2.26 `virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl ( const std::string & brokerUploadUrl ) [virtual]`
- 6.151.2.27 `virtual void activemq::commands::BrokerInfo::setBrokerURL ( const std::string & brokerURL ) [virtual]`
- 6.151.2.28 `virtual void activemq::commands::BrokerInfo::setConnectionId ( long long connectionId ) [virtual]`
- 6.151.2.29 `virtual void activemq::commands::BrokerInfo::setDuplexConnection ( bool duplexConnection ) [virtual]`
- 6.151.2.30 `virtual void activemq::commands::BrokerInfo::setFaultTolerantConfiguration ( bool faultTolerantConfiguration ) [virtual]`

6.151.2.31 `virtual void activemq::commands::BrokerInfo::setMasterBroker ( bool masterBroker ) [virtual]`

6.151.2.32 `virtual void activemq::commands::BrokerInfo::setNetworkConnection ( bool networkConnection ) [virtual]`

6.151.2.33 `virtual void activemq::commands::BrokerInfo::setNetworkProperties ( const std::string & networkProperties ) [virtual]`

6.151.2.34 `virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos ( const std::vector< decaf::lang::Pointer< BrokerInfo > > & peerBrokerInfos ) [virtual]`

6.151.2.35 `virtual void activemq::commands::BrokerInfo::setSlaveBroker ( bool slaveBroker ) [virtual]`

6.151.2.36 `virtual std::string activemq::commands::BrokerInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.151.2.37 `virtual Pointer<Command> activemq::commands::BrokerInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.151.3 Field Documentation

6.151.3.1 `Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId [protected]`

- 6.151.3.2 `std::string activemq::commands::BrokerInfo::brokerName`  
[protected]
- 6.151.3.3 `std::string activemq::commands::BrokerInfo::brokerUploadUrl`  
[protected]
- 6.151.3.4 `std::string activemq::commands::BrokerInfo::brokerURL`  
[protected]
- 6.151.3.5 `long long activemq::commands::BrokerInfo::connectionId`  
[protected]
- 6.151.3.6 `bool activemq::commands::BrokerInfo::duplexConnection`  
[protected]
- 6.151.3.7 `bool activemq::commands::BrokerInfo::faultTolerantConfiguration`  
[protected]
- 6.151.3.8 `const unsigned char activemq::commands::BrokerInfo::ID_BROKERINFO = 2` [static]
- 6.151.3.9 `bool activemq::commands::BrokerInfo::masterBroker` [protected]
- 6.151.3.10 `bool activemq::commands::BrokerInfo::networkConnection`  
[protected]
- 6.151.3.11 `std::string activemq::commands::BrokerInfo::networkProperties`  
[protected]
- 6.151.3.12 `std::vector< decaf::lang::Pointer<BrokerInfo> >`  
`activemq::commands::BrokerInfo::peerBrokerInfos` [protected]
- 6.151.3.13 `bool activemq::commands::BrokerInfo::slaveBroker` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerInfo.h`

## 6.152 `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 862).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller`:

## Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual ~**BrokerInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.152.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 862).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.152.2 Constructor & Destructor Documentation

6.152.2.1 **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::BrokerInfoMarshaller**  
( ) [inline]

6.152.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

### 6.152.3 Member Function Documentation

6.152.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.152.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.152.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).



## 6.152 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference 867

6.152.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.152.3.5 virtual int activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

6.152.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 734).

6.152.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 736).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h`

## 6.153 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 867).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.153.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 867).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.153.2 Constructor & Destructor Documentation

6.153.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

6.153.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

### 6.153.3 Member Function Documentation

6.153.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.153.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.153.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.153.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

6.153.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.153.3.6  virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.153.3.7  virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**BrokerInfoMarshaller.h**

## 6.154 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 871).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.154.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 871).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.154.2 Constructor & Destructor Documentation

6.154.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

6.154.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

### 6.154.3 Member Function Documentation

6.154.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.154.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.154.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## 6.154 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference 875

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.154.3.4  virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.154.3.5  virtual int activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.154.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.154.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerInfoMarshaller.h**

## 6.155 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 875).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.155.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 875).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.155.2 Constructor & Destructor Documentation

6.155.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

6.155.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

### 6.155.3 Member Function Documentation

6.155.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.155.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.155.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 751).

6.155.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 752).

6.155.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.155.3.6  virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.155.3.7  virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**BrokerInfoMarshaller.h**

## 6.156 activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 879).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.156.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 879).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.156.2 Constructor & Destructor Documentation

6.156.2.1 `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

6.156.2.2 `virtual activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

## 6.156.3 Member Function Documentation

6.156.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.156.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.156.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.156.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.156.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.156.3.6  virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.156.3.7  virtual void activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**BrokerInfoMarshaller.h**

## 6.157 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 883).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller:

### Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.157.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 883).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.157.2 Constructor & Destructor Documentation

6.157.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::BrokerInfoMarshaller ( ) [inline]`

6.157.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::~~BrokerInfoMarshaller ( ) [inline, virtual]`

## 6.157.3 Member Function Documentation

6.157.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.157.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.157.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.157.3.4 **virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseUnmarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
[virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

6.157.3.5 **virtual int activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal1**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.157.3.6  virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.157.3.7  virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerInfoMarshaller.h**

## 6.158 decaf::nio::Buffer Class Reference

A container for data of a specific primitive type.

```
#include <src/main/decaf/nio/Buffer.h>
```

Inheritance diagram for decaf::nio::Buffer:

### Public Member Functions

- **Buffer** (int capacity)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()
- virtual int **capacity** () const
- virtual int **position** () const
- virtual **Buffer** & **position** (int newPosition) throw ( lang::exceptions::IllegalArgumentException )  
*Sets this buffer's position.*
- virtual int **limit** () const
- virtual **Buffer** & **limit** (int newLimit) throw ( lang::exceptions::IllegalArgumentException )  
*Sets this buffer's limit.*
- virtual **Buffer** & **mark** ()  
*Sets this buffer's mark at its position.*
- virtual **Buffer** & **reset** () throw ( InvalidMarkException )  
*Resets this buffer's position to the previously-marked position.*
- virtual **Buffer** & **clear** ()  
*Clears this buffer.*
- virtual **Buffer** & **flip** ()  
*Flips this buffer.*
- virtual **Buffer** & **rewind** ()  
*Rewinds this buffer.*
- virtual int **remaining** () const  
*Returns the number of elements between the current position and the limit.*
- virtual bool **hasRemaining** () const  
*Tells whether there are any elements between the current position and the limit.*
- virtual bool **isReadOnly** () const =0  
*Tells whether or not this buffer is read-only.*

## Protected Attributes

- `int _position`
- `int _capacity`
- `int _limit`
- `int _mark`
- `bool _markSet`

### 6.158.1 Detailed Description

A container for data of a specific primitive type.

A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: \* Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p. 916) and a relative put operation throws a **BufferOverflowException** (p. 914); in either case, no data is transferred. \* Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an **IndexOutOfBoundsException** if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p. 2096) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values:  $0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of



the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

**clear()** (p. 890) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

**flip()** (p. 890) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

**rewind()** (p. 893) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 3115) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its `isReadOnly` method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

```
b.flip(); b.position(23); b.limit(42);
```

can be replaced by the single, more compact statement `b.flip().position(23).limit(42);`

## 6.158.2 Constructor & Destructor Documentation

6.158.2.1 `decaf::nio::Buffer::Buffer ( int capacity )`

6.158.2.2 `decaf::nio::Buffer::Buffer ( const Buffer & other )`

6.158.2.3 `virtual decaf::nio::Buffer::~~Buffer ( ) [inline, virtual]`

## 6.158.3 Member Function Documentation

**6.158.3.1** `virtual int decaf::nio::Buffer::capacity ( ) const [inline, virtual]`

#### Returns

this buffer's capacity.

**6.158.3.2** `virtual Buffer& decaf::nio::Buffer::clear ( ) [virtual]`

Clears this buffer.

The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

```
buf.clear(); // Prepare buffer for reading in.read(buf); // Read data
```

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

#### Returns

a reference to this buffer.

**6.158.3.3** `virtual Buffer& decaf::nio::Buffer::flip ( ) [virtual]`

Flips this buffer.

The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

```
buf.put(magic); // Prepend header in.read(buf); // Read data into rest of buffer buf.flip();  
// Flip buffer out.write(buf); // Write header + data to channel
```

This method is often used in conjunction with the compact method when transferring data from one place to another.

#### Returns

a reference to this buffer.

**6.158.3.4** `virtual bool decaf::nio::Buffer::hasRemaining ( ) const [inline, virtual]`

Tells whether there are any elements between the current position and the limit.

#### Returns

true if, and only if, there is at least one element remaining in this buffer.

6.158.3.5 virtual bool decaf::nio::Buffer::isReadOnly ( ) const [pure virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 976), **decaf::internal::nio::CharArrayBuffer** (p. 1086), **decaf::internal::nio::DoubleArrayBuffer** (p. 1771), **decaf::internal::nio::FloatArrayBuffer** (p. 1885), **decaf::internal::nio::IntArrayBuffer** (p. 2024), **decaf::internal::nio::LongArrayBuffer** (p. 2401), **decaf::internal::nio::ShortArrayBuffer** (p. 3399), and **decaf::nio::ByteBuffer** (p. 1012).

6.158.3.6 virtual int decaf::nio::Buffer::limit ( ) const [inline, virtual]

Returns

this buffers Limit

6.158.3.7 virtual Buffer& decaf::nio::Buffer::limit ( int newLimit ) throw ( lang::exceptions::IllegalArgumentException ) [virtual]

Sets this buffer's limit.

If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

Parameters

<i>newLimit</i>	The new limit value; must be no larger than this buffer's capacity.
-----------------	---

Returns

A reference to This buffer

Exceptions

<i>IllegalArgumentException</i>	if preconditions on the new pos don't hold.
---------------------------------	---

6.158.3.8 virtual Buffer& decaf::nio::Buffer::mark ( ) [virtual]

Sets this buffer's mark at its position.

Returns

a reference to this buffer.

6.158.3.9 `virtual int decaf::nio::Buffer::position ( ) const` `[inline, virtual]`

#### Returns

the current position in the buffer

6.158.3.10 `virtual Buffer& decaf::nio::Buffer::position ( int newPosition ) throw ( lang::exceptions::IllegalArgumentException )` `[virtual]`

Sets this buffer's position.

If the mark is defined and larger than the new position then it is discarded.

#### Parameters

<i>newPosition</i>	The new position in the buffer to set.
--------------------	--

#### Returns

a reference to This buffer.

#### Exceptions

<i>IllegalArgumentException</i>	if preconditions on the new pos don't hold.
---------------------------------	---

6.158.3.11 `virtual int decaf::nio::Buffer::remaining ( ) const` `[inline, virtual]`

Returns the number of elements between the current position and the limit.

#### Returns

The number of elements remaining in this buffer

6.158.3.12 `virtual Buffer& decaf::nio::Buffer::reset ( ) throw ( InvalidMarkException )` `[virtual]`

Resets this buffer's position to the previously-marked position.

#### Returns

a reference to this buffer.

#### Exceptions

<i>InvalidMarkException</i> (p. 2096)	- If the mark has not been set
--	--------------------------------

6.158.3.13 virtual **Buffer&** decaf::nio::Buffer::rewind ( ) [virtual]

Rewinds this buffer.

The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

```
out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); //
Copy data into array
```

**Returns**

a reference to this buffer.

## 6.158.4 Field Documentation

## 6.158.4.1 int decaf::nio::Buffer::\_capacity [protected]

## 6.158.4.2 int decaf::nio::Buffer::\_limit [protected]

## 6.158.4.3 int decaf::nio::Buffer::\_mark [protected]

## 6.158.4.4 bool decaf::nio::Buffer::\_markSet [protected]

## 6.158.4.5 int decaf::nio::Buffer::\_position [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**Buffer.h**

## 6.159 decaf::io::BufferedInputStream Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

```
#include <src/main/decaf/io/BufferedInputStream.h>
```

Inheritance diagram for decaf::io::BufferedInputStream:

**Public Member Functions**

- **BufferedInputStream** (**InputStream** \*stream, bool own=false)

*Constructor.*

- **BufferedInputStream** (**InputStream** \*stream, int bufferSize, bool own=false) throw ( lang::exceptions::IllegalArgumentException )

*Constructor.*

- virtual ~**BufferedInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

#### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.*

*The default implementation of this method does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

#### Parameters

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------

#### Returns

*total bytes skipped*

#### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

UnsupportedOperationException	<i>if the concrete stream class does not support skipping bytes.</i>
-------------------------------	--

- virtual void **mark** (int readLimit)

*Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

**Parameters**

readLimit	The max bytes read before marked position is invalid.
-----------	---

- virtual void **reset** () throw ( decaf::io::IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2103).*

**Exceptions**

<b>IOException</b> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

**Returns**

*true if this stream instance supports marks*

**Protected Member Functions**

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**6.159.1 Detailed Description**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

**6.159.2 Constructor & Destructor Documentation**

6.159.2.1 `decaf::io::BufferedInputStream::BufferedInputStream ( InputStream * stream, bool own = false )`

Constructor.

#### Parameters

<i>stream</i>	The target input stream to buffer.
<i>own</i>	Indicates if we own the stream object, defaults to false.

6.159.2.2 `decaf::io::BufferedInputStream::BufferedInputStream ( InputStream * stream, int bufferSize, bool own = false ) throw ( lang::exceptions::IllegalArgumentException )`

Constructor.

#### Parameters

<i>stream</i>	The target input stream to buffer.
<i>bufferSize</i>	The size in bytes to allocate for the internal buffer.
<i>own</i>	Indicates if we own the stream object, defaults to false.

#### Exceptions

<i>IllegalArgumentException</i>	is the size is zero or negative.
---------------------------------	----------------------------------

6.159.2.3 `virtual decaf::io::BufferedInputStream::~~BufferedInputStream ( ) [virtual]`

### 6.159.3 Member Function Documentation

6.159.3.1 `virtual int decaf::io::BufferedInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<i>IOException</i> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------



Reimplemented from **decaf::io::FilterInputStream** (p. 1857).

6.159.3.2 `virtual void decaf::io::BufferedInputStream::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::FilterInputStream** (p. 1857).

6.159.3.3 `virtual int decaf::io::BufferedInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.159.3.4 `virtual int decaf::io::BufferedInputStream::doReadByte ( ) throw ( decaf::io::IOException ) [protected, virtual]`

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.159.3.5 `virtual void decaf::io::BufferedInputStream::mark ( int readLimit ) [virtual]`

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.159.3.6 `virtual bool decaf::io::BufferedInputStream::markSupported ( ) const [inline, virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

### Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1859).

6.159.3.7 **virtual void decaf::io::BufferedInputStream::reset ( ) throw ( decaf::io::IOException )** [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::FilterInputStream** (p. 1859).

6.159.3.8 **virtual long long decaf::io::BufferedInputStream::skip ( long long num ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )** [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number

of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<b>UnsupportedOperationException</b>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::FilterInputStream** (p. 1860).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedInputStream.h**

## 6.160 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

```
#include <src/main/decaf/io/BufferedOutputStream.h>
```

Inheritance diagram for decaf::io::BufferedOutputStream:

#### Public Member Functions

- **BufferedOutputStream** (**OutputStream** \*stream, bool **own**=false)  
*Constructor.*
- **BufferedOutputStream** (**OutputStream** \*stream, int bufferSize, bool **own**=false)  
throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Constructor.*
- virtual ~**BufferedOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )  
*inheritDoc*

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.160.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

### 6.160.2 Constructor & Destructor Documentation

6.160.2.1 **decaf::io::BufferedOutputStream::BufferedOutputStream** ( **OutputStream** \* *stream*, **bool** *own* = *false* )

Constructor.

#### Parameters

<i>stream</i>	The target output stream.
<i>own</i>	Indicates if this class owns the stream pointer.

6.160.2.2 **decaf::io::BufferedOutputStream::BufferedOutputStream** ( **OutputStream** \* *stream*, **int** *bufferSize*, **bool** *own* = *false* ) throw ( **decaf::lang::exceptions::IllegalArgumentException** )

Constructor.

#### Parameters

<i>stream</i>	The target output stream.
<i>bufferSize</i>	The size for the internal buffer.
<i>own</i>	Indicates if this class owns the stream pointer.

#### Exceptions

<i>IllegalArgumentException</i>	if the bufferSize given is negative.
---------------------------------	--------------------------------------

6.160.2.3 **virtual decaf::io::BufferedOutputStream::~~BufferedOutputStream** ( )  
[virtual]

### 6.160.3 Member Function Documentation

6.160.3.1 virtual void decaf::io::BufferedOutputStream::doWriteArray ( const unsigned char \* *buffer*, int *size* ) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.160.3.2 virtual void decaf::io::BufferedOutputStream::doWriteArrayBounded ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.160.3.3 virtual void decaf::io::BufferedOutputStream::doWriteByte ( unsigned char *c* ) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.160.3.4 virtual void decaf::io::BufferedOutputStream::flush ( ) throw ( decaf::io::IOException ) [virtual]

inheritDoc}

Reimplemented from **decaf::io::FilterOutputStream** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedOutputStream.h**

## 6.161 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p. 136) package to create the various default version of the NIO interfaces.

```
#include <src/main/decaf/internal/nio/BufferFactory.h>
```

### Public Member Functions

- virtual ~**BufferFactory** ()

### Static Public Member Functions

- static **decaf::nio::ByteBuffer** \* **createByteBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::ByteBuffer \* createByteBuffer** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new ByteBuffer.*

- static **decaf::nio::ByteBuffer \* createByteBuffer** (std::vector< unsigned char > &buffer)

*Wraps the passed STL Byte Vector in a ByteBuffer.*

- static **decaf::nio::CharBuffer \* createCharBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::CharBuffer \* createCharBuffer** (char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new CharBuffer.*

- static **decaf::nio::CharBuffer \* createCharBuffer** (std::vector< char > &buffer)

*Wraps the passed STL Byte Vector in a CharBuffer.*

- static **decaf::nio::DoubleBuffer \* createDoubleBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::DoubleBuffer \* createDoubleBuffer** (double \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new DoubleBuffer.*

- static **decaf::nio::DoubleBuffer \* createDoubleBuffer** (std::vector< double > &buffer)

*Wraps the passed STL Double Vector in a DoubleBuffer.*

- static **decaf::nio::FloatBuffer \* createFloatBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::FloatBuffer \* createFloatBuffer** (float \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new FloatBuffer.*

- static **decaf::nio::FloatBuffer \* createFloatBuffer** (std::vector< float > &buffer)

*Wraps the passed STL Float Vector in a FloatBuffer.*

- static **decaf::nio::LongBuffer \* createLongBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.*

- static **decaf::nio::LongBuffer \* createLongBuffer** (long long \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new LongBuffer.*
- static **decaf::nio::LongBuffer \* createLongBuffer** (std::vector< long long > &buffer)  
*Wraps the passed STL Long Vector in a LongBuffer.*
- static **decaf::nio::IntBuffer \* createIntBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **decaf::nio::IntBuffer \* createIntBuffer** (int \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new IntBuffer.*
- static **decaf::nio::IntBuffer \* createIntBuffer** (std::vector< int > &buffer)  
*Wraps the passed STL int Vector in a IntBuffer.*
- static **decaf::nio::ShortBuffer \* createShortBuffer** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **decaf::nio::ShortBuffer \* createShortBuffer** (short \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new ShortBuffer.*
- static **decaf::nio::ShortBuffer \* createShortBuffer** (std::vector< short > &buffer)  
*Wraps the passed STL Short Vector in a ShortBuffer.*

### 6.161.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p. 136) package to create the various default version of the NIO interfaces.

Since

1.0

### 6.161.2 Constructor & Destructor Documentation

6.161.2.1 **virtual decaf::internal::nio::BufferFactory::~BufferFactory** ( ) [inline, virtual]

### 6.161.3 Member Function Documentation

```

6.161.3.1 static decaf::nio::ByteBuffer* de-
caf::internal::nio::BufferFactory::createByteBuffer ( int capacity
) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
[static]

```

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated ByteBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

```

6.161.3.2 static decaf::nio::ByteBuffer* de-
caf::internal::nio::BufferFactory::createByteBuffer ( unsigned
char * buffer, int size, int offset, int length ) throw
( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

```

Wraps the passed buffer with a new ByteBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new ByteBuffer that is backed by buffer, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.



6.161.3.3 **static decaf::nio::ByteBuffer\*** decaf::internal::nio::BufferFactory::createByteBuffer ( **std::vector< unsigned char > & buffer** ) **[static]**

Wraps the passed STL Byte Vector in a ByteBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new ByteBuffer that is backed by `buffer`, caller owns.

6.161.3.4 **static decaf::nio::CharBuffer\*** decaf::internal::nio::BufferFactory::createCharBuffer ( **int capacity** ) **throw ( decaf::lang::exceptions::IndexOutOfBoundsException )** **[static]**

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated CharBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.161.3.5 **static decaf::nio::CharBuffer\*** decaf::internal::nio::BufferFactory::createCharBuffer ( **char \* buffer, int size, int offset, int length** ) **throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )** **[static]**

Wraps the passed buffer with a new CharBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer

will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new CharBuffer that is backed by buffer, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.6** `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer ( std::vector< char > & buffer ) [static]`

Wraps the passed STL Byte Vector in a CharBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new CharBuffer that is backed by buffer, caller owns.

**6.161.3.7** `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer ( double * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new DoubleBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer

will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new `DoubleBuffer` that is backed by `buffer`, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

**6.161.3.8** `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer ( std::vector< double > & buffer ) [static]`

Wraps the passed STL Double Vector in a `DoubleBuffer`.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new `DoubleBuffer` that is backed by `buffer`, caller owns.

**6.161.3.9** `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters**

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

**Returns**

a newly allocated DoubleBuffer which the caller owns.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

```
6.161.3.10 static decaf::nio::FloatBuffer* de-
caf::internal::nio::BufferFactory::createFloatBuffer ( int capacity
) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
[static]
```

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters**

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

**Returns**

a newly allocated FloatBuffer which the caller owns.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

```
6.161.3.11 static decaf::nio::FloatBuffer* de-
caf::internal::nio::BufferFactory::createFloatBuffer ( float
* buffer, int size, int offset, int length ) throw (
decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]
```

Wraps the passed buffer with a new FloatBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.

<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

**Returns**

a new FloatBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions**

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.161.3.12 **static decaf::nio::FloatBuffer\*** decaf::internal::nio::BufferFactory::createFloatBuffer (   
std::vector< float > & *buffer* ) [static]

Wraps the passed STL Float Vector in a FloatBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new FloatBuffer that is backed by buffer, caller owns.

6.161.3.13 **static decaf::nio::IntBuffer\*** decaf::internal::nio::BufferFactory::createIntBuffer ( int *capacity* )   
throw ( decaf::lang::exceptions::IndexOutOfBoundsException )   
[static]

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

**Parameters**

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

**Returns**

a newly allocated IntBuffer which the caller owns.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

**6.161.3.14** `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer ( std::vector< int > & buffer ) [static]`

Wraps the passed STL int Vector in a IntBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

**Returns**

a new IntBuffer that is backed by buffer, caller owns.

**6.161.3.15** `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer ( int * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new IntBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

**Returns**

a new IntBuffer that is backed by buffer, caller owns the returned pointer.

**Exceptions**

<i>NullPointerException</i>	if the buffer given in Null.
-----------------------------	------------------------------

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

6.161.3.16 **static decaf::nio::LongBuffer\*** decaf::internal::nio::BufferFactory::createLongBuffer (   
std::vector< long long > & *buffer* ) [static]

Wraps the passed STL Long Vector in a LongBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

#### Returns

a new LongBuffer that is backed by buffer, caller owns.

6.161.3.17 **static decaf::nio::LongBuffer\*** decaf::internal::nio::BufferFactory::createLongBuffer ( int *capacity* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	- the internal buffer's capacity.
-----------------	-----------------------------------

#### Returns

a newly allocated DoubleBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--

```

6.161.3.18 static decaf::nio::LongBuffer* de-
caf::internal::nio::BufferFactory::createLongBuffer ( long
long * buffer, int size, int offset, int length ) throw (
decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

```

Wraps the passed buffer with a new LongBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new LongBuffer that is backed by `buffer`, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

```

6.161.3.19 static decaf::nio::ShortBuffer* de-
caf::internal::nio::BufferFactory::createShortBuffer ( int
capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
[static]

```

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated ShortBuffer which the caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.
----------------------------------	--



6.161.3.20 **static decaf::nio::ShortBuffer\*** decaf::internal::nio::BufferFactory::createShortBuffer ( short \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]

Wraps the passed buffer with a new ShortBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The array that will back the new buffer.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new ShortBuffer that is backed by buffer, caller owns the returned pointer.

#### Exceptions

<i>NullPointerException</i>	if the buffer given in Null.
<i>IndexOutOfBoundsException</i>	if the capacity specified is negative.

6.161.3.21 **static decaf::nio::ShortBuffer\*** decaf::internal::nio::BufferFactory::createShortBuffer ( std::vector< short > & *buffer* ) [static]

Wraps the passed STL Short Vector in a ShortBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

#### Returns

a new DoubleBuffer that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**BufferFactory.h**

## 6.162 decaf::nio::BufferOverflowException Class Reference

```
#include <src/main/decaf/nio/BufferOverflowException.h>
```

Inheritance diagram for decaf::nio::BufferOverflowException:

### Public Member Functions

- **BufferOverflowException** () throw ()  
*Default Constructor.*
- **BufferOverflowException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **BufferOverflowException** (const **BufferOverflowException** &ex) throw ()  
*Copy Constructor.*
- **BufferOverflowException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BufferOverflowException** (const std::exception \*cause) throw ()  
*Constructor.*
- **BufferOverflowException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **BufferOverflowException** \* clone () const  
*Clones this exception.*
- virtual ~**BufferOverflowException** () throw ()

### 6.162.1 Constructor & Destructor Documentation

6.162.1.1 decaf::nio::BufferOverflowException::BufferOverflowException ( ) throw ()  
[inline]

Default Constructor.

6.162.1.2 decaf::nio::BufferOverflowException::BufferOverflowException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

#### Parameters

ex	the exception to copy
----	-----------------------

6.162.1.3 `decaf::nio::BufferOverflowException::BufferOverflowException ( const BufferOverflowException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.162.1.4 `decaf::nio::BufferOverflowException::BufferOverflowException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.162.1.5 `decaf::nio::BufferOverflowException::BufferOverflowException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.162.1.6 `decaf::nio::BufferOverflowException::BufferOverflowException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.162.1.7 `virtual decaf::nio::BufferOverflowException::~~BufferOverflowException ( ) throw ()`  
`[inline, virtual]`

## 6.162.2 Member Function Documentation

6.162.2.1 `virtual BufferOverflowException* decaf::nio::BufferOverflowException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferOverflowException.h`

## 6.163 decaf::nio::BufferUnderflowException Class Reference

```
#include <src/main/decaf/nio/BufferUnderflowException.h>
```

Inheritance diagram for `decaf::nio::BufferUnderflowException`:

### Public Member Functions

- **BufferUnderflowException** ( ) throw ( )  
*Default Constructor.*
- **BufferUnderflowException** (const **lang::Exception** &ex) throw ( )  
*Copy Constructor.*
- **BufferUnderflowException** (const **BufferUnderflowException** &ex) throw ( )  
*Copy Constructor.*
- **BufferUnderflowException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ( )  
*Constructor - Initializes the file name and line number where this message occurred.*
- **BufferUnderflowException** (const std::exception \*cause) throw ( )  
*Constructor.*
- **BufferUnderflowException** (const char \*file, const int lineNumber, const char \*msg,...) throw ( )  
*Constructor.*
- virtual **BufferUnderflowException** \* **clone** ( ) const  
*Clones this exception.*
- virtual ~**BufferUnderflowException** ( ) throw ( )

### 6.163.1 Constructor & Destructor Documentation

6.163.1.1 `decaf::nio::BufferUnderflowException::BufferUnderflowException ( ) throw ()`  
[inline]

Default Constructor.

6.163.1.2 `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const lang::Exception & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.163.1.3 `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const BufferUnderflowException & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.163.1.4 `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.163.1.5 `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const std::exception * cause ) throw ()` [inline]

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.163.1.6 `decaf::nio::BufferUnderflowException::BufferUnderflowException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor.

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.163.1.7 `virtual decaf::nio::BufferUnderflowException::~BufferUnderflowException ( ) throw ()` `[inline, virtual]`

**6.163.2 Member Function Documentation**

6.163.2.1 `virtual BufferUnderflowException* decaf::nio::BufferUnderflowException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferUnderflowException.h`

**6.164 decaf::lang::Byte Class Reference**

```
#include <src/main/decaf/lang/Byte.h>
```

Inheritance diagram for `decaf::lang::Byte`:

**Public Member Functions**

- **Byte** (unsigned char value)

- **Byte** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Byte** ()
- virtual int **compareTo** (const **Byte** &c) const  
*Compares this **Byte** (p. 918) instance with another.*
- virtual bool **operator==** (const **Byte** &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Byte** &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const unsigned char &c) const  
*Compares this **Byte** (p. 918) instance with a char type.*
- virtual bool **operator==** (const unsigned char &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const unsigned char &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

### Static Public Member Functions

- static std::string **toString** (unsigned char value)
- static **Byte decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3610) into a **Byte** (p. 918).*
- static unsigned char **parseByte** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed unsigned char in the radix specified by the second argument.*
- static unsigned char **parseByte** (const std::string &s) throw ( exceptions::NumberFormatException )

*Parses the string argument as a signed decimal unsigned char.*

- static **Byte valueOf** (unsigned char value)

*Returns a **Character** (p. 1069) instance representing the specified char value.*

- static **Byte valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )

*Returns a **Byte** (p. 918) object holding the value given by the specified std::string.*

- static **Byte valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )

*Returns a **Byte** (p. 918) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

### Static Public Attributes

- static const unsigned char **MIN\_VALUE** = 0x7F

*The minimum value that a unsigned char can take on.*

- static const unsigned char **MAX\_VALUE** = 0x80

*The maximum value that a unsigned char can take on.*

- static const int **SIZE** = 8

*The size of the primitive charactor in bits.*

### 6.164.1 Constructor & Destructor Documentation

#### 6.164.1.1 decaf::lang::Byte::Byte ( unsigned char value )

##### Parameters

<i>value</i>	- the primitive value to wrap
--------------	-------------------------------

#### 6.164.1.2 decaf::lang::Byte::Byte ( const std::string & value ) throw ( exceptions::NumberFormatException )

##### Parameters

<i>value</i>	- the string to convert to an unsigned char
--------------	---

##### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

#### 6.164.1.3 virtual decaf::lang::Byte::~~Byte ( ) [inline, virtual]

### 6.164.2 Member Function Documentation



6.164.2.1 `virtual unsigned char decaf::lang::Byte::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

#### Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2787).

6.164.2.2 `virtual int decaf::lang::Byte::compareTo ( const unsigned char & c ) const [inline, virtual]`

Compares this **Byte** (p. 918) instance with a char type.

#### Parameters

<code>c</code>	- the char instance to be compared
----------------	------------------------------------

#### Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **unsigned char** > (p. 1187).

6.164.2.3 `virtual int decaf::lang::Byte::compareTo ( const Byte & c ) const [inline, virtual]`

Compares this **Byte** (p. 918) instance with another.

#### Parameters

<code>c</code>	- the <b>Byte</b> (p. 918) instance to be compared
----------------	--

#### Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Byte** > (p. 1187).

6.164.2.4 `static Byte decaf::lang::Byte::decode ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Decodes a **String** (p. 3610) into a **Byte** (p. 918).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p. 925) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a `NumberFormatException` will be thrown. The result is negated if first character of the specified **String** (p. 3610) is the minus sign. No whitespace characters are permitted in the string.

#### Parameters

<i>value</i>	- The string to decode
--------------	------------------------

#### Returns

a **Byte** (p. 918) object containing the decoded value

#### Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

6.164.2.5 `virtual double decaf::lang::Byte::doubleValue ( ) const [inline, virtual]`

Answers the double value which the receiver represents.

#### Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.164.2.6 `bool decaf::lang::Byte::equals ( const unsigned char & c ) const [inline, virtual]`

#### Returns

true if the two Bytes have the same value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1188).

6.164.2.7 `bool decaf::lang::Byte::equals ( const Byte & c ) const [inline, virtual]`

#### Returns

true if the two **Byte** (p. 918) Objects have the same value.

Implements **decaf::lang::Comparable< Byte >** (p. 1188).

6.164.2.8 virtual float decaf::lang::Byte::floatValue ( ) const [inline, virtual]

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.164.2.9 virtual int decaf::lang::Byte::intValue ( ) const [inline, virtual]

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.164.2.10 virtual long long decaf::lang::Byte::longValue ( ) const [inline, virtual]

Answers the long value which the receiver represents.

#### Returns

long long the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.164.2.11 virtual bool decaf::lang::Byte::operator< ( const unsigned char & c ) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>c</i> - the value to be compared to this one.
--

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p. 1188).

6.164.2.12 `virtual bool decaf::lang::Byte::operator< ( const Byte & c ) const` `[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Byte** > (p. 1188).

6.164.2.13 `virtual bool decaf::lang::Byte::operator==( const Byte & c ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Byte** > (p. 1189).

6.164.2.14 `virtual bool decaf::lang::Byte::operator==( const unsigned char & c ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **unsigned char** > (p. 1189).

6.164.2.15 static unsigned char decaf::lang::Byte::parseByte ( const std::string & s ) throw ( exceptions::NumberFormatException ) [static]

Parses the string argument as a signed decimal unsigned char.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseByte( const std::string, int ) method.

#### Parameters

s	- <b>String</b> (p. 3610) to convert to a unsigned char
---	---

#### Returns

the converted unsigned char value

#### Exceptions

<i>NumberFormatException</i>	if the string is not a unsigned char.
------------------------------	---------------------------------------

6.164.2.16 static unsigned char decaf::lang::Byte::parseByte ( const std::string & s, int radix ) throw ( exceptions::NumberFormatException ) [static]

Parses the string argument as a signed unsigned char in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1072) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type NumberFormatException is thrown if any of the following situations occurs: \* The first argument is null or is a string of length zero. \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1076) or larger than **Character::MAX\_RADIX** (p. 1076). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type unsigned char.

#### Parameters

s	- the <b>String</b> (p. 3610) containing the unsigned char to be parsed
radix	- the radix to be used while parsing s

#### Returns

the unsigned char represented by the string argument in the specified radix.

#### Exceptions

<i>NumberFormatException</i>	- If <b>String</b> (p. 3610) does not contain a parsable unsigned char.
------------------------------	---

6.164.2.17 `virtual short decaf::lang::Byte::shortValue ( ) const [inline, virtual]`

Answers the short value which the receiver represents.

#### Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2788).

6.164.2.18 `static std::string decaf::lang::Byte::toString ( unsigned char value ) [static]`

#### Returns

a string representing the primitive value as Base 10

6.164.2.19 `std::string decaf::lang::Byte::toString ( ) const`

#### Returns

this **Byte** (p. 918) Object as a **String** (p. 3610) Representation

6.164.2.20 `static Byte decaf::lang::Byte::valueOf ( unsigned char value ) [inline, static]`

Returns a **Character** (p. 1069) instance representing the specified char value.

#### Parameters

<i>value</i>	- the primitive char to wrap.
--------------	-------------------------------

#### Returns

a new **Character** (p. 1069) instance that wraps this value.

6.164.2.21 `static Byte decaf::lang::Byte::valueOf ( const std::string & value, int radix ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Byte** (p. 918) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the `parseByte( std::string, int )` method. The result is a **Byte** (p.918) object that represents the unsigned char value specified by the string.

#### Parameters

<i>value</i>	- std::string to parse as base ( radix )
<i>radix</i>	- base of the string to parse.

#### Returns

new **Byte** (p.918) Object wrapping the primitive

#### Exceptions

<i>NumberFormatException</i>	if the string is not a valid unsigned char.
------------------------------	---

**6.164.2.22** `static Byte decaf::lang::Byte::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Byte** (p.918) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the `parseByte( std::string )` method. The result is a **Byte** (p.918) object that represents the unsigned char value specified by the string.

#### Parameters

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

#### Returns

new **Byte** (p.918) Object wrapping the primitive

#### Exceptions

<i>NumberFormatException</i>	if the string is not a decimal unsigned char.
------------------------------	---

### 6.164.3 Field Documentation

**6.164.3.1** `const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80 [static]`

The maximum value that a unsigned char can take on.

6.164.3.2 `const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F` [static]

The minimum value that a unsigned char can take on.

6.164.3.3 `const int decaf::lang::Byte::SIZE = 8` [static]

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Byte.h`

## 6.165 decaf::internal::util::ByteArrayAdapter Class Reference

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

```
#include <src/main/decaf/internal/util/ByteArrayAdapter.h>
```

### Data Structures

- union **Array**

### Public Member Functions

- **ByteArrayAdapter** (int size) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.*
- **ByteArrayAdapter** (unsigned char \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (char \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (double \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (float \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (long long \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*



- **ByteArrayAdapter** (int \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- **ByteArrayAdapter** (short \*array, int size, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte array object that wraps the given array.*
- virtual ~**ByteArrayAdapter** ()
- virtual int **getCapacity** () const  
*Gets the size of the underlying array.*
- virtual int **getCharCapacity** () const  
*Gets the size of the underlying array as if it contains chars.*
- virtual int **getDoubleCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getFloatCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getLongCapacity** () const  
*Gets the size of the underlying array as if it contains doubles.*
- virtual int **getIntCapacity** () const  
*Gets the size of the underlying array as if it contains ints.*
- virtual int **getShortCapacity** () const  
*Gets the size of the underlying array as if it contains shorts.*
- virtual unsigned char \* **getByteArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual char \* **getCharArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual short \* **getShortArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual int \* **getIntArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual long long \* **getLongArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual double \* **getDoubleArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual float \* **getFloatArray** ()  
*Gets the pointer to the array we are wrapping.*
- virtual void **read** (unsigned char \*buffer, int size, int offset, int length) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException )  
*Reads from the Byte array starting at the specified offset and reading the specified length.*
- virtual void **write** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException )

*Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.*

- virtual void **resize** (int size) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InvalidStateException )

*Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.*

- virtual void **clear** () throw ()

*Clear all data from that Array, setting the underlying bytes to zero.*

- unsigned char & **operator[]** (int index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allows the **ByteArrayAdapter** (p. 928) to be indexed as a standard array.*

- const unsigned char & **operator[]** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

- virtual unsigned char **get** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

- virtual char **getChar** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads one byte at the given index and returns it.*

- virtual double **getDouble** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

- virtual double **getDoubleAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given byte index and returns it.*

- virtual float **getFloat** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

- virtual float **getFloatAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given byte index and returns it.*

- virtual long long **getLong** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

- virtual long long **getLongAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given byte index and returns it.*

- virtual int **getInt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

- virtual int **getIntAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given byte index and returns it.*

- virtual short **getShort** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads two bytes at the given index and returns it.*

- virtual short **getShortAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads two bytes at the given byte index and returns it.*

- virtual **ByteArrayAdapter** & **put** (int index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes the given byte into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putChar** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes one byte containing the given value, into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putDouble** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putDoubleAt** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes eight bytes containing the given value, into this buffer at the given byte index.*

- virtual **ByteArrayAdapter** & **putFloat** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes four bytes containing the given value, into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putFloatAt** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes four bytes containing the given value, into this buffer at the given byte index.*

- virtual **ByteArrayAdapter** & **putLong** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putLongAt** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes eight bytes containing the given value, into this buffer at the given byte index.*

- virtual **ByteArrayAdapter** & **putInt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes four bytes containing the given value, into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putIntAt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes four bytes containing the given value, into this buffer at the given byte index.*

- virtual **ByteArrayAdapter** & **putShort** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes two bytes containing the given value, into this buffer at the given index.*

- virtual **ByteArrayAdapter** & **putShortAt** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Writes two bytes containing the given value, into this buffer at the given byte index.*

### 6.165.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

#### Since

1.0

### 6.165.2 Constructor & Destructor Documentation

#### 6.165.2.1 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( int size ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
-------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if size is negative.
---------------------------------	----------------------

#### 6.165.2.2 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( unsigned char * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.3** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( char * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

**6.165.2.4** `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( double * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.165.2.5 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( float * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.165.2.6 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( long long * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.165.2.7 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( int * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.165.2.8 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter ( short * array, int size, bool own = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The physical array to wrap.
<i>size</i>	The size of the array, this is the limit we read and write to.
<i>own</i>	Indicates if this class is now the owner of the pointer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the size is negative.

6.165.2.9 `virtual decaf::internal::util::ByteArrayAdapter::~~ByteArrayAdapter ( )`  
[virtual]

**6.165.3 Member Function Documentation**

6.165.3.1 `virtual void decaf::internal::util::ByteArrayAdapter::clear ( ) throw ()` [virtual]

Clear all data from that Array, setting the underlying bytes to zero.

6.165.3.2 `virtual unsigned char decaf::internal::util::ByteArrayAdapter::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Absolute get method.

Reads the byte at the given index.

**Parameters**

<i>index</i>	The index in the Buffer where the byte is to be read.
--------------	---

**Returns**

the byte that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	If index is not smaller than the buffer's limit or is negative.
----------------------------------	---

6.165.3.3 `virtual unsigned char* decaf::internal::util::ByteArrayAdapter::getByteArray ( )`  
`[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 928) objects that point to this array.

**Returns**

an unsigned char\* pointer to the array this object wraps.

6.165.3.4 `virtual int decaf::internal::util::ByteArrayAdapter::getCapacity ( ) const`  
`[inline, virtual]`

Gets the size of the underlying array.

**Returns**

the size the array.

6.165.3.5 `virtual char decaf::internal::util::ByteArrayAdapter::getChar ( int index ) const throw (`  
`decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Reads one byte at the given index and returns it.

**Parameters**

<i>index</i>	The index in the Buffer where the byte is to be read.
--------------	---

**Returns**

the byte that is located at the given index.

**Exceptions**



<i>IndexOutOfBoundsException</i>	If index is not smaller than the buffer's limit or is negative.
----------------------------------	---

**6.165.3.6** `virtual char* decaf::internal::util::ByteArrayAdapter::getCharArray ( ) [inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 928) objects that point to this array.

#### Returns

an char\* pointer to the array this object wraps.

**6.165.3.7** `virtual int decaf::internal::util::ByteArrayAdapter::getCharCapacity ( ) const [inline, virtual]`

Gets the size of the underlying array as if it contains chars.

#### Returns

the size the array.

**6.165.3.8** `virtual double decaf::internal::util::ByteArrayAdapter::getDouble ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

#### Returns

the value at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.9 `virtual double* decaf::internal::util::ByteArrayAdapter::getDoubleArray ( )`  
`[inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 928) objects that point to this array.

#### Returns

an `double*` pointer to the array this object wraps.

6.165.3.10 `virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Reads eight bytes at the given byte index and returns it.

#### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

#### Returns

the value at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.11 `virtual int decaf::internal::util::ByteArrayAdapter::getDoubleCapacity ( ) const`  
`[inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

#### Returns

the size the array.

6.165.3.12 `virtual float decaf::internal::util::ByteArrayAdapter::getFloat ( int index ) const`  
`throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.13 `virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray ( )`  
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 928) objects that point to this array.

**Returns**

an float\* pointer to the array this object wraps.

6.165.3.14 `virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Reads four bytes at the given byte index and returns it.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.15 `virtual int decaf::internal::util::ByteArrayAdapter::getFloatCapacity ( ) const`  
[inline, virtual]

Gets the size of the underlying array as if it contains doubles.

**Returns**

the size the array.

6.165.3.16 `virtual int decaf::internal::util::ByteArrayAdapter::getInt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

**Returns**

the value at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.17 `virtual int* decaf::internal::util::ByteArrayAdapter::getIntArray ( )` [inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 928) objects that point to this array.

**Returns**

an int\* pointer to the array this object wraps.

6.165.3.18 `virtual int decaf::internal::util::ByteArrayAdapter::getIntAt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads four bytes at the given byte index and returns it.

**Parameters**

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

**Returns**

the value at the given index in the buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.19 `virtual int decaf::internal::util::ByteArrayAdapter::getIntCapacity ( ) const`  
[inline, virtual]

Gets the size of the underlying array as if it contains ints.

### Returns

the size the array.

6.165.3.20 `virtual long long decaf::internal::util::ByteArrayAdapter::getLong ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

### Returns

the value at the given index in the buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.21 `virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray ( )`  
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 928) objects that point to this array.

### Returns

an long long\* pointer to the array this object wraps.

6.165.3.22 `virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt ( int index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads eight bytes at the given byte index and returns it.

#### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

#### Returns

the value at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.23 `virtual int decaf::internal::util::ByteArrayAdapter::getLongCapacity ( ) const  
[inline, virtual]`

Gets the size of the underlying array as if it contains doubles.

#### Returns

the size the array.

6.165.3.24 `virtual short decaf::internal::util::ByteArrayAdapter::getShort ( int index ) const  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]`

Reads two bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read.
--------------	---

#### Returns

the value at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.25 `virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray ( )`  
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 928) objects that point to this array.

#### Returns

an short\* pointer to the array this object wraps.

6.165.3.26 `virtual short decaf::internal::util::ByteArrayAdapter::getShortAt ( int index )`  
`const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Reads two bytes at the given byte index and returns it.

#### Parameters

<i>index</i>	The index in the Buffer where the bytes are to be read
--------------	--

#### Returns

the value at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

6.165.3.27 `virtual int decaf::internal::util::ByteArrayAdapter::getShortCapacity ( ) const`  
[inline, virtual]

Gets the size of the underlying array as if it contains shorts.

#### Returns

the size the array.

6.165.3.28 `unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] ( int index ) throw`  
`( decaf::lang::exceptions::IndexOutOfBoundsException )`

Allows the **ByteArrayAdapter** (p. 928) to be indexed as a standard array.

calling the non constant version allows the user to change the value at index

#### Parameters

<i>index</i>	The position in the array to access, if the value is negative or greater than the size of the underlying array an <code>IndexOutOfBoundsException</code> is thrown.
--------------	---

### Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions of <code>index</code> are not met.
----------------------------------	---

6.165.3.29 `const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] ( int index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`

6.165.3.30 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put  
( int index, unsigned char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Writes the given byte into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

### Returns

a reference to this buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if <code>index</code> greater than the buffer's limit minus the size of the type being written, or <code>index</code> is negative.
----------------------------------	--

6.165.3.31 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putChar  
( int index, char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
[virtual]

Writes one byte containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.



### Returns

a reference to this buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.32 **virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDouble**  
( int *index*, double *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

### Returns

a reference to this buffer.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.33 **virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDoubleAt**  
( int *index*, double *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

### Returns

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.34 **virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat**  
 ( int *index*, float *value* ) throw ( de-  
 caf::lang::exceptions::IndexOutOfBoundsException )  
 [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

**Parameters**

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.35 **virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt**  
 ( int *index*, float *value* ) throw ( de-  
 caf::lang::exceptions::IndexOutOfBoundsException )  
 [virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

**Parameters**

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

```
6.165.3.36 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt  
          ( int index, int value ) throw ( de-  
          caf::lang::exceptions::IndexOutOfBoundsException )  
          [virtual]
```

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

```
6.165.3.37 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt  
          ( int index, int value ) throw ( de-  
          caf::lang::exceptions::IndexOutOfBoundsException )  
          [virtual]
```

Writes four bytes containing the given value, into this buffer at the given byte index.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.38 **virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong**  
( int *index*, long long *value* ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.39 **virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt**  
( int *index*, long long *value* ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.40 virtual **ByteArrayAdapter**& decaf::internal::util::ByteArrayAdapter::putShort  
( int *index*, short *value* ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes two bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof( type )` if the actual start index of the type to be read.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write to the array.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.41 virtual **ByteArrayAdapter**& decaf::internal::util::ByteArrayAdapter::putShortAt  
( int *index*, short *value* ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

#### Parameters

<i>index</i>	The position in the Buffer to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
----------------------------------	--

6.165.3.42 `virtual void decaf::internal::util::ByteArrayAdapter::read ( unsigned char * buffer, int size, int offset, int length ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException ) [virtual]`

Reads from the Byte array starting at the specified offset and reading the specified length.

If the length is greater than the size of this underlying byte array then an `BufferUnderflowException` is thrown.

#### Parameters

<i>buffer</i>	The buffer to read data from this array into.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in this array to start reading from.
<i>length</i>	The amount of data to read from this array.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length exceeds the size.
<i>NullPointerException</i>	if buffer is null
<i>BufferUnderflowException</i>	if there is not enough data to read because the offset or the length is greater than the size of this array.

6.165.3.43 `virtual void decaf::internal::util::ByteArrayAdapter::resize ( int size ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InvalidStateException ) [virtual]`

Resizes the underlying array to the new given size, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.

A **ByteArrayAdapter** (p. 928) can only be resized when it owns the underlying array, if it does not then it will throw an `InvalidStateException`.

#### Parameters

<i>size</i>	The new size of the array.
-------------	----------------------------

#### Exceptions

<i>IllegalArgumentException</i>	if the size parameter is negative.
<i>InvalidStateException</i>	if this object does not own the buffer.

6.165.3.44 virtual void decaf::internal::util::ByteArrayAdapter::write ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException ) [virtual]

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.

. If the length is greater than the size of this underlying byte array then an BufferOverflowException is thrown.

#### Parameters

<i>buffer</i>	The buffer to read data from this array into.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in this array to start reading from.
<i>length</i>	The amount of data to read from this array.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length exceeds the size.
<i>NullPointerException</i>	if buffer is null
<i>BufferOverflowException</i>	if the amount of data to be written to this array or the offset given are larger than this array's size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ByteBufferAdapter.h**

## 6.166 decaf::internal::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/internal/nio/ByteBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ByteBuffer:

#### Public Member Functions

- **ByteBuffer** (int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
Creates a **ByteBuffer** (p. 951) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.
- **ByteBuffer** (unsigned char \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **ByteBuffer** (p. 951) object that wraps the given array.

- **ByteBuffer** (const **decaf::lang::Pointer**< **ByteBufferAdapter** > &array, int offset, int length, bool readOnly=false) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException** )

Creates a byte buffer that wraps the passed **ByteBufferAdapter** and start at the given offset.

- **ByteBuffer** (const **ByteBuffer** &other)

Create a **ByteBuffer** (p. 951) that mirrors this one, meaning it shares a reference to this buffers **ByteBufferAdapter** and when changes are made to that data it is reflected in both.

- virtual ~**ByteBuffer** ()
- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

#### Returns

true if, and only if, this buffer is read-only

- virtual unsigned char \* **array** () throw ( **decaf::nio::ReadOnlyBufferException**, **decaf::lang::exceptions::UnsupportedOperationException** )

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The array that backs this buffer

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is backed by an array but is read-only
<b>UnsupportedOperationException</b>	if this buffer is not backed by an accessible array

- virtual int **arrayOffset** () const throw ( **decaf::nio::ReadOnlyBufferException**, **decaf::lang::exceptions::UnsupportedOperationException** )

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position *p* corresponds to array index *p* + **arrayOffset()** (p. 1001).

Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset within this buffer's array of the first element of the buffer.

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is backed by an array but is read-only.
<b>UnsupportedOperationException</b>	if this buffer is not backed by an accessible array.

- virtual bool **hasArray** () const



*Tells whether or not this buffer is backed by an accessible byte array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

#### Returns

*true if, and only if, this buffer is backed by an array and is not read-only.*

- virtual **decaf::nio::CharBuffer \* asCharBuffer ()** const

*Creates a view of this byte buffer as a char buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*the new Char **Buffer** (p. 887), which the caller then owns.*

- virtual **decaf::nio::DoubleBuffer \* asDoubleBuffer ()** const

*Creates a view of this byte buffer as a double buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*the new double **Buffer** (p. 887), which the caller then owns.*

- virtual **decaf::nio::FloatBuffer \* asFloatBuffer ()** const

*Creates a view of this byte buffer as a float buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*the new float **Buffer** (p. 887), which the caller then owns.*

- virtual **decaf::nio::IntBuffer \* asIntBuffer ()** const

*Creates a view of this byte buffer as a int buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*the new int **Buffer** (p. 887), which the caller then owns.*

- virtual **decaf::nio::LongBuffer \* asLongBuffer ()** const

*Creates a view of this byte buffer as a long buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*the new long **Buffer** (p. 887), which the caller then owns.*

- virtual **decaf::nio::ShortBuffer \* asShortBuffer ()** const

*Creates a view of this byte buffer as a short buffer.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*the new short **Buffer** (p. 887), which the caller then owns.*

- virtual **ByteBuffer \* asReadOnlyBuffer ()** const

*Creates a new, read-only byte buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.*

*If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.*

#### Returns

*The new, read-only byte buffer which the caller then owns.*

- virtual **ByteBuffer & compact ()** throw ( decaf::nio::ReadOnlyBufferException )

*Compacts this buffer.*

*The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.*

*The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.*

#### Returns

*a reference to this **ByteBuffer** (p. 995).*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

- virtual **ByteBuffer \* duplicate ()**

*Creates a new byte buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.*

#### Returns

*a new Byte **Buffer** (p. 887) which the caller owns.*

- virtual unsigned char **get** () const throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the byte at this buffer's current position, and then increments the position.*

#### Returns

*The byte at the buffer's current position.*

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	<i>if the buffer's current position is not smaller than its limit.</i>
---	--

- virtual unsigned char **get** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the byte at the given index.*

#### Parameters

index	<i>The index in the <b>Buffer</b> (p. 887) where the byte is to be read.</i>
-------	--

#### Returns

*the byte that is located at the given index.*

#### Exceptions

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual char **getChar** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next byte at this buffer's current position, and then increments the position by one.*

#### Returns

*the next char in the buffer.*

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
---	--

- virtual char **getChar** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads one byte at the given index and returns it.*

#### Parameters

index	<i>The index in the <b>Buffer</b> (p. 887) where the byte is to be read.</i>
-------	--

**Returns**

*the char at the given index in the buffer*

**Exceptions**

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual double **getDouble** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next double in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
---	--

- virtual double **getDouble** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.</i>
-------	--

**Returns**

*the double at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual float **getFloat** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next float in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
---	--

- virtual float **getFloat** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.</i>
-------	--

**Returns**

*the float at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---------------------------	--

- virtual long long **getLong** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next long long in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

- virtual long long **getLong** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads eight bytes at the given index and returns it.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
-------	---

**Returns**

*the long long at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---------------------------	--

- virtual int **getInt** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next int in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

- virtual int **getInt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads four bytes at the given index and returns it.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
-------	---

**Returns**

*the int at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
---------------------------	--

- virtual short **getShort** () throw ( decaf::nio::BufferUnderflowException )

*Reads the next two bytes at this buffer's current position, and then increments the position by that amount.*

**Returns**

*the next short in the buffer.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	<i>if there are no more bytes remaining in this buffer, meaning we have reached the set limit.</i>
---	--

- virtual short **getShort** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads two bytes at the given index and returns it.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.</i>
-------	--

**Returns**

*the short at the given index in the buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if there are not enough bytes remaining to fill the requested Data Type, or index is negative.</i>
---------------------------	---

- virtual **ByteBuffer & put** (unsigned char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given byte into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>- the byte value to be written.</i>
-------	--

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	<i>if this buffer's current position is not smaller than its limit.</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only.</i>

- virtual **ByteBuffer & put** (int index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given byte into this buffer at the given index.*

**Parameters**

index	<i>- position in the <b>Buffer</b> (p. 887) to write the data</i>
value	<i>- the byte to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer & putChar** (char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

- virtual **ByteBuffer & putChar** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data.
value	The value to write.

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

- virtual **ByteBuffer & putDouble** (double value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

- virtual **ByteBuffer & putDouble** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data
value	The value to write.

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>IndexOutOfBoundsException</b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer & putFloat** (float value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer & putFloat** (int index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data
value	The value to write.

**Returns**

*a reference to this buffer.*



**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer & putLong** (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer & putLong** (int index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes eight bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data.
value	The value to write.

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer & putInt** (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

- virtual **ByteBuffer & putInt** (int index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException  
decaf::nio::ReadOnlyBufferException )

*Writes four bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data.
value	The value to write.

**Returns**

*a reference to this buffer*

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

- virtual **ByteBuffer & putShort** (short value) throw ( decaf::nio::BufferOverflowException,  
decaf::nio::ReadOnlyBufferException )

*Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

**Parameters**

value	The value to be written.
-------	--------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer & putShort** (int index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException  
decaf::nio::ReadOnlyBufferException )

*Writes two bytes containing the given value, into this buffer at the given index.*

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data
value	The value to write.

**Returns**

*a reference to this buffer*

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual **ByteBuffer \* slice ()** const

*Creates a new byte buffer whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the newly create **ByteBuffer** (p. 995) which the caller owns.*

**Protected Member Functions**

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 951) as Read-Only or not Read-Only.*

**6.166.1 Detailed Description**

This class defines six categories of operations upon byte buffers:

1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 973) float **getFloat(int index)** void **putFloat(float f)** (p. 980) void **putFloat(int index, float f)** (p. 979)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The `asFloatBuffer` method, for example, creates an instance of the `FloatBuffer` class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

#### Since

1.0

### 6.166.2 Constructor & Destructor Documentation

6.166.2.1 `decaf::internal::nio::ByteBuffer::ByteBuffer ( int capacity, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **ByteBuffer** (p. 951) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Should this buffer be read-only, default as false

#### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

6.166.2.2 `decaf::internal::nio::ByteBuffer::ByteBuffer ( unsigned char * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **ByteBuffer** (p. 951) object that wraps the given array.

#### Parameters

<i>array</i>	The array to wrap.
<i>size</i>	The size of the array passed.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The size of the sub-array, this is the limit we read and write to.
<i>readOnly</i>	Should this buffer be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset and length are violated.

6.166.2.3 `decaf::internal::nio::ByteBuffer::ByteBuffer ( const decaf::lang::Pointer< ByteBufferAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteBufferAdapter and start at the given offset.

The capacity and limit of the new **ByteBuffer** (p. 951) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteBufferAdapter to wrap
<i>offset</i>	The offset into array where the buffer starts
<i>length</i>	The length of the array we are wrapping or limit.
<i>readOnly</i>	Boolean indicating if this a readOnly buffer.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.166.2.4 `decaf::internal::nio::ByteBuffer::ByteBuffer ( const ByteBuffer & other )`

Create a **ByteBuffer** (p. 951) that mirrors this one, meaning it shares a reference to this buffers `ByteBufferAdapter` and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>ByteBuffer</b> (p. 951) this one is to mirror.
--------------	---

6.166.2.5 `virtual decaf::internal::nio::ByteBuffer::~~ByteBuffer ( ) [virtual]`

### 6.166.3 Member Function Documentation

6.166.3.1 `virtual unsigned char* decaf::internal::nio::ByteBuffer::array ( ) throw ( decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The array that backs this buffer

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is backed by an array but is read-only
<b><i>UnsupportedOperationException</i></b>	if this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p. 1001).

6.166.3.2 `virtual int decaf::internal::nio::ByteBuffer::arrayOffset ( ) const throw ( decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position `p` corresponds to array index `p +`

**arrayOffset()** (p. 1001).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset within this buffer's array of the first element of the buffer.

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is backed by an array but is read-only.
<b><i>UnsupportedOperationException</i></b>	if this buffer is not backed by an accessible array.

Implements **decaf::nio::ByteBuffer** (p. 1001).

**6.166.3.3** virtual **decaf::nio::CharBuffer\*** **decaf::internal::nio::ByteBuffer::asCharBuffer** ( ) const  
[inline, virtual]

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new Char **Buffer** (p. 887), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1002).

**6.166.3.4** virtual **decaf::nio::DoubleBuffer\*** **decaf::internal::nio::ByteBuffer::asDoubleBuffer** ( ) const  
[inline, virtual]

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the new double **Buffer** (p. 887), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1002).

```
6.166.3.5  virtual decaf::nio::FloatBuffer* de-
           caf::internal::nio::ByteBuffer::asFloatBuffer (    ) const
           [inline, virtual]
```

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the new float **Buffer** (p. 887), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1002).

```
6.166.3.6  virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteBuffer::asIntBuffer (    )
           const [inline, virtual]
```

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the new int **Buffer** (p. 887), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1003).

```
6.166.3.7  virtual decaf::nio::LongBuffer* de-
           caf::internal::nio::ByteBuffer::asLongBuffer (    ) const
           [inline, virtual]
```

Creates a view of this byte buffer as a long buffer.



The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new long **Buffer** (p. 887), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1003).

6.166.3.8 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::asReadOnlyBuffer ( ) const [virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only byte buffer which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1003).

6.166.3.9 `virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteBuffer::asShortBuffer ( ) const [inline, virtual]`

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new short **Buffer** (p. 887), which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 1004).

6.166.3.10 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::compact ( )`  
`throw ( decaf::nio::ReadOnlyBufferException )` [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **ByteBuffer** (p. 995).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::ByteBuffer** (p. 1004).

6.166.3.11 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::duplicate ( )`  
[virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new **Byte Buffer** (p. 887) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 1005).

6.166.3.12 `virtual unsigned char decaf::internal::nio::ByteBuffer::get ( ) const` `throw ( decaf::nio::BufferUnderflowException )` [virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

### Returns

The byte at the buffer's current position.

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if the buffer's current position is not smaller than its limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 1006).

```
6.166.3.13  virtual unsigned char decaf::internal::nio::ByteBuffer::get ( int index )
              const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
              [virtual]
```

Absolute get method.

Reads the byte at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the byte is to be read.
--------------	---

### Returns

the byte that is located at the given index.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1006).

```
6.166.3.14  virtual char decaf::internal::nio::ByteBuffer::getChar ( ) throw (
              decaf::nio::BufferUnderflowException ) [inline, virtual]
```

Reads the next byte at this buffer's current position, and then increments the position by one.

### Returns

the next char in the buffer.

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 1007).

```
6.166.3.15 virtual char decaf::internal::nio::ByteBuffer::getChar ( int index ) const throw (
    decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
    virtual]
```

Reads one byte at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the byte is to be read.
--------------	---

#### Returns

the char at the given index in the buffer

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1007).

```
6.166.3.16 virtual double decaf::internal::nio::ByteBuffer::getDouble ( ) throw (
    decaf::nio::BufferUnderflowException ) [virtual]
```

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next double in the buffer.

#### Exceptions

<i>BufferUnderflowException</i> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 1008).

```
6.166.3.17 virtual double decaf::internal::nio::ByteBuffer::getDouble ( int index )
    const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
    [virtual]
```

Reads eight bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

**Returns**

the double at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1008).

6.166.3.18 virtual float decaf::internal::nio::ByteBuffer::getFloat ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

**Returns**

the next float in the buffer.

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 1009).

6.166.3.19 virtual float decaf::internal::nio::ByteBuffer::getFloat ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads four bytes at the given index and returns it.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

**Returns**

the float at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1009).

6.166.3.20 `virtual int decaf::internal::nio::ByteBuffer::getInt ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next int in the buffer.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 1009).

6.166.3.21 `virtual int decaf::internal::nio::ByteBuffer::getInt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Reads four bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

#### Returns

the int at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1010).

6.166.3.22 `virtual long long decaf::internal::nio::ByteBuffer::getLong ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next long long in the buffer.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implements **decaf::nio::ByteBuffer** (p. 1010).

6.166.3.23 virtual long long decaf::internal::nio::ByteBuffer::getLong ( int *index* )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Reads eight bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

#### Returns

the long long at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1011).

6.166.3.24 virtual short decaf::internal::nio::ByteBuffer::getShort ( int *index* ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads two bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

#### Returns

the short at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implements **decaf::nio::ByteBuffer** (p. 1011).

6.166.3.25 `virtual short decaf::internal::nio::ByteBuffer::getShort ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]`

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next short in the buffer.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implements **decaf::nio::ByteBuffer** (p. 1011).

6.166.3.26 `virtual bool decaf::internal::nio::ByteBuffer::hasArray ( ) const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ByteBuffer** (p. 1012).

6.166.3.27 `virtual bool decaf::internal::nio::ByteBuffer::isReadOnly ( ) const [inline, virtual]`

Tells whether or not this buffer is read-only.

#### Returns

true if, and only if, this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1012).

6.166.3.28 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put ( unsigned char value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given byte into this buffer at the current position, and then increments the position.



**Parameters**

<i>value</i>	- the byte value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1012).

```
6.166.3.29 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::put
( int index, unsigned char value ) throw (
    decaf::lang::exceptions::IndexOutOfBoundsException,
    decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes the given byte into this buffer at the given index.

**Parameters**

<i>index</i>	- position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	- the byte to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1013).

```
6.166.3.30 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar
( int index, char value ) throw ( de-
    caf::lang::exceptions::IndexOutOfBoundsException,
    decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes one byte containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1016).

```
6.166.3.31 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar
( char value ) throw ( decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1015).

```
6.166.3.32 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble
( int index, double value ) throw ( de-
caaf::lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1017).

6.166.3.33 virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putDouble  
( double *value* ) throw ( decaf::nio::BufferOverflowException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1016).

6.166.3.34 virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putFloat  
( int *index*, float *value* ) throw ( de-  
cafe::lang::exceptions::IndexOutOfBoundsException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1018).

```
6.166.3.35 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat
( float value ) throw ( decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1017).

```
6.166.3.36 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt ( int index,
int value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes four bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1018).

6.166.3.37 virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putInt  
( int *value* ) throw ( **decaf::nio::BufferOverflowException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 1019).

6.166.3.38 virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putLong  
( long long *value* ) throw ( **decaf::nio::BufferOverflowException**,  
**decaf::nio::ReadOnlyBufferException** ) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1019).

```
6.166.3.39 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong
( int index, long long value ) throw ( de-
caf::lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes eight bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The value to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1020).

```
6.166.3.40 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort
( short value ) throw ( decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

**Parameters**

<i>value</i>	The value to be written.
--------------	--------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1021).

6.166.3.41 virtual **ByteBuffer&** decaf::internal::nio::ByteBuffer::putShort  
( int *index*, short *value* ) throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Writes two bytes containing the given value, into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The value to write.

**Returns**

a reference to this buffer

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ByteBuffer** (p. 1020).

6.166.3.42 virtual void decaf::internal::nio::ByteBuffer::setReadOnly ( bool *value* )  
[inline, protected, virtual]

Sets this **ByteBuffer** (p. 951) as Read-Only or not Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.166.3.43 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::slice ( ) const`  
`[virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the newly create **ByteBuffer** (p. 995) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 1021).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteBuffer.h`

**6.167 decaf::io::ByteArrayInputStream Class Reference**

A **ByteArrayInputStream** (p. 984) contains an internal buffer that contains bytes that may be read from the stream.

```
#include <src/main/decaf/io/ByteArrayInputStream.h>
```

Inheritance diagram for `decaf::io::ByteArrayInputStream`:

**Public Member Functions**

- **ByteArrayInputStream ()**  
*Creates an **ByteArrayInputStream** (p. 984) with an empty input buffer, the buffer can be initialized with a call to `setByteArray`.*
- **ByteArrayInputStream (const std::vector< unsigned char > &buffer)**  
*Creates the input stream and calls `setBuffer` with the specified buffer object.*
- **ByteArrayInputStream (const unsigned char \*buffer, int bufferSize, bool own=false)**  
`throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )`



Create an instance of the **ByteArrayInputStream** (p. 984) with the given buffer as the source of input for all read operations.

- **ByteArrayInputStream** (const unsigned char \*buffer, int bufferSize, int offset, int length, bool own=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

Create an instance of the **ByteArrayInputStream** (p. 984) with the given buffer as the source of input for all read operations.

- virtual ~**ByteArrayInputStream** ()
- virtual void **setByteArray** (const std::vector< unsigned char > &buffer)  
Sets the internal buffer.
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual void **setByteArray** (const unsigned char \*buffer, int bufferSize, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual int **available** () const throw ( IOException )  
Indicates the number of bytes available.  
The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.  
The default implementation of this method returns zero.

**Returns**  
the number of bytes available on this input stream.

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------

- virtual long long **skip** (long long num) throw ( io::IOException, lang::exceptions::UnsupportedOperationException )

Skips over and discards n bytes of data from this input stream.  
The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.  
The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

#### Returns

total bytes skipped

**Exceptions**

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
UnsupportedOperationException	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)

*Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*

*If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.*

*Calling mark on a closed stream instance should have no effect.*

*The default implementation of this method does nothing.*

**Parameters**

readLimit	<i>The max bytes read before marked position is invalid.</i>
-----------	--

- virtual void **reset** () throw ( IOException )

*Repositions this stream to the position at the time the mark method was last called on this input stream.*

*If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.*

*If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.*

*The default implementation of this method throws an **IOException** (p. 2103).*

**Exceptions**

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual bool **markSupported** () const

*Determines if this input stream supports the mark and reset methods.*

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

**Returns**

*true if this stream instance supports marks*

**Protected Member Functions**

- virtual int **doReadByte** () throw ( IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.167.1 Detailed Description

A **ByteArrayInputStream** (p. 984) contains an internal buffer that contains bytes that may be read from the stream.

An internal counter keeps track of the next byte to be supplied by the read method. The **ByteArrayInputStream** (p. 984) never copies the supplied buffers, only points to them, therefore the caller must ensure that the supplied buffer remain in scope, or is not deleted before this **ByteArrayInputStream** (p. 984) is freed. If the own argument of one of the constructors that accepts an array pointer is set to true than the **ByteArrayInputStream** (p. 984) instance will take ownership of the supplied pointer and delete it when that instance is destroyed.

Closing a **ByteArrayInputStream** (p. 984) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p. 2103).

#### Since

1.0

### 6.167.2 Constructor & Destructor Documentation

#### 6.167.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( )

Creates an **ByteArrayInputStream** (p. 984) with an empty input buffer, the buffer can be initialized with a call to `setByteArray`.

#### 6.167.2.2 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( const std::vector< unsigned char > & *buffer* )

Creates the input stream and calls `setBuffer` with the specified buffer object.

#### Parameters

<i>buffer</i>	The buffer to be used.
---------------	------------------------

#### 6.167.2.3 decaf::io::ByteArrayInputStream::ByteArrayInputStream ( const unsigned char \* *buffer*, int *bufferSize*, bool *own* = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )

Create an instance of the **ByteArrayInputStream** (p. 984) with the given buffer as the source of input for all read operations.

#### Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.

<i>own</i>	Indicates if this object should take ownership of the array, default is false.
------------	--

### Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx- ception</i>	if the bufferSize is negative.

```
6.167.2.4  decaf::io::ByteArrayInputStream::ByteArrayInputStream ( const unsigned
char * buffer, int bufferSize, int offset, int length, bool own = false
) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException )
```

Create an instance of the **ByteArrayInputStream** (p. 984) with the given buffer as the source of input for all read operations.

### Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>offset</i>	The offset into the buffer to begin reading from.
<i>length</i>	The number of bytes to read past the offset.
<i>own</i>	Indicates if this object should take ownership of the array, default is false.

### Exceptions

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx- ception</i>	if the bufferSize is negative.

```
6.167.2.5  virtual decaf::io::ByteArrayInputStream::~~ByteArrayInputStream ( )
[virtual]
```

## 6.167.3 Member Function Documentation

```
6.167.3.1  virtual int decaf::io::ByteArrayInputStream::available ( ) const throw ( IOException
) [virtual]
```

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns**

the number of bytes available on this input stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2004).

**6.167.3.2** `virtual int decaf::io::ByteArrayInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2005).

**6.167.3.3** `virtual int decaf::io::ByteArrayInputStream::doReadByte ( ) throw ( IOException )` [protected, virtual]

Implements **decaf::io::InputStream** (p. 2005).

**6.167.3.4** `virtual void decaf::io::ByteArrayInputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

**Parameters**

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::InputStream** (p. 2006).

**6.167.3.5** `virtual bool decaf::io::ByteArrayInputStream::markSupported ( ) const` [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular

input stream instance.

The default implementation of this method returns false.

### Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 2006).

**6.167.3.6** `virtual void decaf::io::ByteArrayInputStream::reset ( ) throw ( IOException )`  
`[virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2009).

**6.167.3.7** `virtual void decaf::io::ByteArrayInputStream::setByteArray ( const unsigned char *  
buffer, int bufferSize ) throw ( decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

### Parameters

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.

**Exceptions**

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx-ception</i>	if the bufferSize is negative.

**6.167.3.8** virtual void decaf::io::ByteArrayInputStream::setByteArray ( const std::vector< unsigned char > & *buffer* ) [virtual]

Sets the internal buffer.

The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

**Parameters**

<i>buffer</i>	The buffer to be used.
---------------	------------------------

**6.167.3.9** virtual void decaf::io::ByteArrayInputStream::setByteArray ( const unsigned char \* *buffer*, int *bufferSize*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

**Parameters**

<i>buffer</i>	The initial byte array to use to read from.
<i>bufferSize</i>	The size of the buffer.
<i>offset</i>	The offset into the buffer to begin reading from.
<i>length</i>	The number of bytes to read past the offset.

**Exceptions**

<i>NullPointerException</i>	if the buffer is Null.
<i>IllegalArgumentEx-ception</i>	if the bufferSize is negative.

**6.167.3.10** virtual long long decaf::io::ByteArrayInputStream::skip ( long long *num* ) throw ( io::IOException, lang::exceptions::UnsupportedOperationException ) [virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller num-

ber of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2010).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**ByteArrayInputStream.h**

## 6.168 decaf::io::ByteArrayOutputStream Class Reference

```
#include <src/main/decaf/io/ByteArrayOutputStream.h>
```

Inheritance diagram for decaf::io::ByteArrayOutputStream:

#### Public Member Functions

- **ByteArrayOutputStream** ()  
*Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.*
- **ByteArrayOutputStream** (int bufferSize) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ByteArrayOutputStream** (p. 992) with an internal buffer allocated with the given size.*
- virtual ~**ByteArrayOutputStream** ()
- std::pair< unsigned char \*, int > **toByteArray** () const  
*Creates a newly allocated byte array.*



- long long **size** () const  
*Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 992).*
- virtual void **reset** () throw ( IOException )  
*Clear current Stream contents.*
- virtual std::string **toString** () const  
*Converts the bytes in the buffer into a standard C++ string.*
- void **writeTo** ( **OutputStream** \*out) const throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write( buf, 0, count ).*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.168.1 Constructor & Destructor Documentation

#### 6.168.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ( )

Default Constructor - uses a default internal buffer of 32 bytes, the size increases as the need for more room arises.

#### 6.168.1.2 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ( int bufferSize ) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **ByteArrayOutputStream** (p. 992) with an internal buffer allocated with the given size.

#### Parameters

<i>bufferSize</i>	The size to use for the internal buffer.
-------------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if the size is less than or equal to zero.
---------------------------------	--

#### 6.168.1.3 virtual decaf::io::ByteArrayOutputStream::~~ByteArrayOutputStream ( ) [virtual]

### 6.168.2 Member Function Documentation

6.168.2.1 `virtual void decaf::io::ByteArrayOutputStream::doWriteArrayBounded  
( const unsigned char * buffer, int size, int offset, int length ) throw (   
decaf::io::IOException, decaf::lang::exceptions::NullPointerException,   
decaf::lang::exceptions::IndexOutOfBoundsException )  
[protected, virtual]`

Reimplemented from **decaf::io::OutputStream** (p. 2859).

6.168.2.2 `virtual void decaf::io::ByteArrayOutputStream::doWriteByte ( unsigned char value )  
throw ( decaf::io::IOException ) [protected, virtual]`

Implements **decaf::io::OutputStream** (p. 2859).

6.168.2.3 `virtual void decaf::io::ByteArrayOutputStream::reset ( ) throw ( IOException )  
[virtual]`

Clear current Stream contents.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)
--

6.168.2.4 `long long decaf::io::ByteArrayOutputStream::size ( ) const`

Gets the current count of bytes written into this **ByteArrayOutputStream** (p. 992).

#### Returns

the number of valid bytes contained in the **ByteArrayOutputStream** (p. 992).

6.168.2.5 `std::pair<unsigned char*, int> decaf::io::ByteArrayOutputStream::toByteArray ( )  
const`

Creates a newly allocated byte array.

Its size is the current size of this output stream and the valid contents of the buffer have been copied into it. The newly allocated array and its size are returned inside an STL pair structure, the caller is responsible for freeing the returned array.

#### Returns

an STL pair containing the copied array and its size.

6.168.2.6 `virtual std::string decaf::io::ByteArrayOutputStream::toString ( ) const`  
`[virtual]`

Converts the bytes in the buffer into a standard C++ string.

#### Returns

a string containing the bytes in the buffer

Reimplemented from **decaf::io::OutputStream** (p. 2860).

6.168.2.7 `void decaf::io::ByteArrayOutputStream::writeTo ( OutputStream * out ) const throw`  
`( decaf::io::IOException, decaf::lang::exceptions::NullPointerException`  
`)`

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using `out.write( buf, 0, count )`.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteBuffer.h`

## 6.169 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/nio/ByteBuffer.h>
```

Inheritance diagram for `decaf::nio::ByteBuffer`:

### Public Member Functions

- `virtual ~ByteBuffer ()`
- `virtual std::string toString () const`
- **ByteBuffer & get** (`std::vector< unsigned char > buffer`) throw ( `BufferUnderflowException` )  
*Relative bulk get method.*
- **ByteBuffer & get** (`unsigned char *buffer`, `int size`, `int offset`, `int length`) throw ( `BufferUnderflowException`, `decaf::lang::exceptions::IndexOutOfBoundsException`, `decaf::lang::exceptions::NullPointerException` )  
*Relative bulk get method.*
- **ByteBuffer & put** (**ByteBuffer &src**) throw ( `BufferOverflowException`, `ReadOnlyBufferException`, `decaf::lang::exceptions::IllegalArgumentException` )  
*This method transfers the bytes remaining in the given source buffer into this buffer.*

- **ByteBuffer & put** (const unsigned char \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers bytes into this buffer from the given source array.*
- **ByteBuffer & put** (std::vector< unsigned char > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source byte array into this buffer.*
- virtual bool **isReadOnly** () const =0  
*Tells whether or not this buffer is read-only.*
- virtual unsigned char \* **array** ()=0 throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns the byte array that backs this buffer.*
- virtual int **arrayOffset** () const =0 throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns the offset within this buffer's backing array of the first element of the buffer.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible byte array.*
- virtual **CharBuffer** \* **asCharBuffer** () const =0  
*Creates a view of this byte buffer as a char buffer.*
- virtual **DoubleBuffer** \* **asDoubleBuffer** () const =0  
*Creates a view of this byte buffer as a double buffer.*
- virtual **FloatBuffer** \* **asFloatBuffer** () const =0  
*Creates a view of this byte buffer as a float buffer.*
- virtual **IntBuffer** \* **asIntBuffer** () const =0  
*Creates a view of this byte buffer as a int buffer.*
- virtual **LongBuffer** \* **asLongBuffer** () const =0  
*Creates a view of this byte buffer as a long buffer.*
- virtual **ShortBuffer** \* **asShortBuffer** () const =0  
*Creates a view of this byte buffer as a short buffer.*
- virtual **ByteBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only byte buffer that shares this buffer's content.*
- virtual **ByteBuffer & compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **ByteBuffer** \* **duplicate** ()=0  
*Creates a new byte buffer that shares this buffer's content.*
- virtual unsigned char **get** () const =0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual unsigned char **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- virtual char **getChar** ()=0 throw ( BufferUnderflowException )  
*Reads the next byte at this buffer's current position, and then increments the position by one.*

- virtual char **getChar** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads one byte at the given index and returns it.*
- virtual double **getDouble** ()=0 throw ( BufferUnderflowException )  
*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*
- virtual double **getDouble** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given index and returns it.*
- virtual float **getFloat** ()=0 throw ( BufferUnderflowException )  
*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*
- virtual float **getFloat** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual long long **getLong** ()=0 throw ( BufferUnderflowException )  
*Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.*
- virtual long long **getLong** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads eight bytes at the given index and returns it.*
- virtual int **getInt** ()=0 throw ( BufferUnderflowException )  
*Reads the next four bytes at this buffer's current position, and then increments the position by that amount.*
- virtual int **getInt** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads four bytes at the given index and returns it.*
- virtual short **getShort** ()=0 throw ( BufferUnderflowException )  
*Reads the next two bytes at this buffer's current position, and then increments the position by that amount.*
- virtual short **getShort** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads two bytes at the given index and returns it.*
- virtual **ByteBuffer** & **put** (unsigned char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given byte into this buffer at the current position, and then increments the position.*
- virtual **ByteBuffer** & **put** (int index, unsigned char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given byte into this buffer at the given index.*
- virtual **ByteBuffer** & **putChar** (char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.*
- virtual **ByteBuffer** & **putChar** (int index, char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )

*Writes one byte containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putDouble** (double value)=0 throw ( `BufferOverflowException`, `ReadOnlyBufferException` )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putDouble** (int index, double value)=0 throw ( `decaf::lang::exceptions::IndexOutOfBoundsException`, `ReadOnlyBufferException` )

*Writes eight bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putFloat** (float value)=0 throw ( `BufferOverflowException`, `ReadOnlyBufferException` )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putFloat** (int index, float value)=0 throw ( `decaf::lang::exceptions::IndexOutOfBoundsException`, `ReadOnlyBufferException` )

*Writes four bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putLong** (long long value)=0 throw ( `BufferOverflowException`, `ReadOnlyBufferException` )

*Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putLong** (int index, long long value)=0 throw ( `decaf::lang::exceptions::IndexOutOfBoundsException`, `ReadOnlyBufferException` )

*Writes eight bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putInt** (int value)=0 throw ( `BufferOverflowException`, `ReadOnlyBufferException` )

*Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putInt** (int index, int value)=0 throw ( `decaf::lang::exceptions::IndexOutOfBoundsException`, `ReadOnlyBufferException` )

*Writes four bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer & putShort** (short value)=0 throw ( `BufferOverflowException`, `ReadOnlyBufferException` )

*Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.*

- virtual **ByteBuffer & putShort** (int index, short value)=0 throw ( `decaf::lang::exceptions::IndexOutOfBoundsException`, `ReadOnlyBufferException` )

*Writes two bytes containing the given value, into this buffer at the given index.*

- virtual **ByteBuffer \* slice** () const =0

*Creates a new byte buffer whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **ByteBuffer** &value) const

- virtual bool **equals** (const **ByteBuffer** &value) const

- virtual bool **operator==** (const **ByteBuffer** &value) const

- virtual bool **operator<** (const **ByteBuffer** &value) const

## Static Public Member Functions

- static **ByteBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.*
- static **ByteBuffer** \* **wrap** (unsigned char \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **ByteBuffer** (p. 995).*
- static **ByteBuffer** \* **wrap** (std::vector< unsigned char > &buffer)  
*Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 995).*

## Protected Member Functions

- **ByteBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ByteBuffer** (p. 995) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.169.1 Detailed Description

This class defines six categories of operations upon byte buffers:

1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 1009) float **getFloat**(int index) void **putFloat**(float f) (p. 1017) void **putFloat**(int index, float f) (p. 1018)

Corresponding methods are defined for the types `char`, `short`, `int`, `long`, and `double`. The index parameters of the absolute `get` and `put` methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The `asFloatBuffer` method, for example, creates an instance of the **FloatBuffer** (p. 1887) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types `char`, `short`, `int`, `long`, and `double`.

View buffers have two important advantages over the families of type-specific `get` and `put` methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk `get` and `put` methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

## 6.169.2 Constructor & Destructor Documentation

6.169.2.1 `decaf::nio::ByteBuffer::ByteBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` `[protected]`

Creates a **ByteBuffer** (p. 995) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
-----------------	--

### Exceptions

<i>IllegalArgumentException</i>	if <i>capacity</i> is negative.
---------------------------------	---------------------------------

6.169.2.2 `virtual decaf::nio::ByteBuffer::~~ByteBuffer ( )` `[inline, virtual]`

## 6.169.3 Member Function Documentation



6.169.3.1 `static ByteBuffer* decaf::nio::ByteBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

#### Parameters

<i>capacity</i>	The internal buffer's capacity.
-----------------	---------------------------------

#### Returns

a newly allocated **ByteBuffer** (p. 995) which the caller owns.

#### Exceptions

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

6.169.3.2 `virtual unsigned char* decaf::nio::ByteBuffer::array ( ) throw ( ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The array that backs this buffer

#### Exceptions

<i>ReadOnlyBufferException</i> (p. 3115)	if this buffer is backed by an array but is read-only
<i>UnsupportedOperationException</i>	if this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 966).

```
6.169.3.3 virtual int decaf::nio::ByteBuffer::arrayOffset ( )
const throw ( ReadOnlyBufferException,
decaf::lang::exceptions::UnsupportedOperationException ) [pure
virtual]
```

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position *p* corresponds to array index *p* + **arrayOffset()** (p. 1001).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset within this buffer's array of the first element of the buffer.

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is backed by an array but is read-only.
<b><i>UnsupportedOperationException</i></b>	if this buffer is not backed by an accessible array.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 966).

```
6.169.3.4 virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer ( ) const [pure
virtual]
```

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the new Char **Buffer** (p. 887), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 967).

```
6.169.3.5 virtual DoubleBuffer* decaf::nio::ByteBuffer::asDoubleBuffer ( ) const [pure
virtual]
```

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new double **Buffer** (p. 887), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 967).

**6.169.3.6** `virtual FloatBuffer* decaf::nio::ByteBuffer::asFloatBuffer ( ) const` [pure virtual]

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new float **Buffer** (p. 887), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 968).

**6.169.3.7** `virtual IntBuffer* decaf::nio::ByteBuffer::asIntBuffer ( ) const` [pure virtual]

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the new int **Buffer** (p. 887), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 968).

**6.169.3.8** `virtual LongBuffer* decaf::nio::ByteBuffer::asLongBuffer ( ) const` `[pure virtual]`

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the new long **Buffer** (p. 887), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 968).

**6.169.3.9** `virtual ByteBuffer* decaf::nio::ByteBuffer::asReadOnlyBuffer ( ) const` `[pure virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 969).

**6.169.3.10** `virtual ShortBuffer* decaf::nio::ByteBuffer::asShortBuffer ( ) const` `[pure virtual]`

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the new short **Buffer** (p. 887), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 969).

**6.169.3.11** virtual **ByteBuffer&** decaf::nio::ByteBuffer::compact ( ) throw ( **ReadOnlyBufferException** ) [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **ByteBuffer** (p. 995).

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implemented in **decaf::internal::nio::ByteBuffer** (p. 970).

**6.169.3.12** virtual int decaf::nio::ByteBuffer::compareTo ( const **ByteBuffer** & *value* ) const [virtual]

**6.169.3.13** virtual **ByteBuffer\*** decaf::nio::ByteBuffer::duplicate ( ) [pure virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new Byte **Buffer** (p. 887) which the caller owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 970).

6.169.3.14 `virtual bool decaf::nio::ByteBuffer::equals ( const ByteBuffer & value ) const`  
`[virtual]`

6.169.3.15 `ByteBuffer& decaf::nio::ByteBuffer::get ( std::vector< unsigned char > buffer )`  
`throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this Byte **Buffer** (p. 887).

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length bytes remaining in this buffer
--	---

6.169.3.16 `virtual unsigned char decaf::nio::ByteBuffer::get ( ) const throw (`  
`BufferUnderflowException ) [pure virtual]`

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

### Returns

The byte at the buffer's current position.

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if the buffer's current position is not smaller than its limit.
--	---

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 970).

6.169.3.17 `virtual unsigned char decaf::nio::ByteBuffer::get ( int index ) const throw (`  
`decaf::lang::exceptions::IndexOutOfBoundsException ) [pure`  
`virtual]`

Absolute get method.

Reads the byte at the given index.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the byte is to be read.
--------------	---

**Returns**

the byte that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 971).

**6.169.3.18 ByteBuffer& decaf::nio::ByteBuffer::get ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

Relative bulk get method.

This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 892), then no bytes are transferred and a **BufferUnderflowException** (p. 916) is thrown.

Otherwise, this method copies `length` bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

**Parameters**

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the passed in <b>Buffer</b> (p. 887).
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

**Returns**

a reference to this **Buffer** (p. 887).

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.
<b>BufferUnderflowException</b> (p. 916)	if there are fewer than length bytes remaining in this buffer.
<i>NullPointerException</i>	if the passed buffer is null.

6.169.3.19 `virtual char decaf::nio::ByteBuffer::getChar ( ) throw ( BufferUnderflowException ) [pure virtual]`

Reads the next byte at this buffer's current position, and then increments the position by one.

#### Returns

the next char in the buffer.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 971).

6.169.3.20 `virtual char decaf::nio::ByteBuffer::getChar ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads one byte at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the byte is to be read.
--------------	---

#### Returns

the char at the given index in the buffer

#### Exceptions

<b><i>IndexOutOfBoundsException</i></b>	if index is not smaller than the buffer's limit, or index is negative.
---	--

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 972).

6.169.3.21 `virtual double decaf::nio::ByteBuffer::getDouble ( ) throw ( BufferUnderflowException ) [pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next double in the buffer.

#### Exceptions



<b><i>BufferUnderflowException</i></b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 972).

6.169.3.22 `virtual double decaf::nio::ByteBuffer::getDouble ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads eight bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

#### Returns

the double at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 972).

6.169.3.23 `virtual float decaf::nio::ByteBuffer::getFloat ( ) throw ( BufferUnderflowException ) [pure virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next float in the buffer.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
--	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 973).

6.169.3.24 `virtual float decaf::nio::ByteBuffer::getFloat ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads four bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

#### Returns

the float at the given index in the buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 973).

6.169.3.25 `virtual int decaf::nio::ByteBuffer::getInt ( ) throw ( BufferUnderflowException )` [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next int in the buffer.

#### Exceptions

<i>BufferUnderflowException</i> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 974).

6.169.3.26 `virtual int decaf::nio::ByteBuffer::getInt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads four bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

**Returns**

the int at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 974).

6.169.3.27 virtual long long decaf::nio::ByteBuffer::getLong ( ) throw ( **BufferUnderflowException** ) [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

**Returns**

the next long long in the buffer.

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteBuffer** (p. 974).

6.169.3.28 virtual long long decaf::nio::ByteBuffer::getLong ( int *index* ) const throw ( **decaf::lang::exceptions::IndexOutOfBoundsException** ) [pure virtual]

Reads eight bytes at the given index and returns it.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

**Returns**

the long long at the given index in the buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteBuffer** (p. 975).

6.169.3.29 `virtual short decaf::nio::ByteBuffer::getShort ( ) throw ( BufferUnderflowException ) [pure virtual]`

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

#### Returns

the next short in the buffer.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there are no more bytes remaining in this buffer, meaning we have reached the set limit.
---	---

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 976).

6.169.3.30 `virtual short decaf::nio::ByteBuffer::getShort ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Reads two bytes at the given index and returns it.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the bytes are to be read.
--------------	---

#### Returns

the short at the given index in the buffer.

#### Exceptions

<b>IndexOutOfBoundsException</b>	if there are not enough bytes remaining to fill the requested Data Type, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 975).

6.169.3.31 `virtual bool decaf::nio::ByteBuffer::hasArray ( ) const [pure virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 976).

6.169.3.32 `virtual bool decaf::nio::ByteBuffer::isReadOnly ( ) const [pure virtual]`

Tells whether or not this buffer is read-only.

### Returns

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 890).

Implemented in **decaf::internal::nio::ByteBuffer** (p. 976).

6.169.3.33 `virtual bool decaf::nio::ByteBuffer::operator< ( const ByteBuffer & value ) const [virtual]`

6.169.3.34 `virtual bool decaf::nio::ByteBuffer::operator== ( const ByteBuffer & value ) const [virtual]`

6.169.3.35 `virtual ByteBuffer& decaf::nio::ByteBuffer::put ( unsigned char value ) throw ( BufferOverflowException, ReadOnlyBufferException ) [pure virtual]`

Writes the given byte into this buffer at the current position, and then increments the position.

### Parameters

<i>value</i>	- the byte value to be written.
--------------	---------------------------------

### Returns

a reference to this buffer.

### Exceptions

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 976).

6.169.3.36 `virtual ByteBuffer& decaf::nio::ByteBuffer::put ( int index, unsigned char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]`

Writes the given byte into this buffer at the given index.

#### Parameters

<i>index</i>	- position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	- the byte to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 977).

6.169.3.37 `ByteBuffer& decaf::nio::ByteBuffer::put ( ByteBuffer & src ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the bytes remaining in the given source buffer into this buffer.

If there are more bytes remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 892), then no bytes are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `n = src.remaining()` bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take bytes from an place in this one.
------------	---

#### Returns

a reference to this buffer

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer for the remaining bytes in the source buffer
--	--

<i>IllegalArgumentException</i>	if the source buffer is this buffer
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

6.169.3.38 **ByteBuffer**& decaf::nio::ByteBuffer::put ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException**, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

This method transfers bytes into this buffer from the given source array.

If there are more bytes to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 892), then no bytes are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The array from which bytes are to be read.
<i>size</i>	The size of the given array.
<i>offset</i>	The offset within the array of the first byte to be read.
<i>length</i>	The number of bytes to be read from the given array.

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.169.3.39 **ByteBuffer&** decaf::nio::ByteBuffer::put ( std::vector< unsigned char > & *buffer* )  
throw ( **BufferOverflowException**, **ReadOnlyBufferException** )

This method transfers the entire content of the given source byte array into this buffer.

This is the same as calling put( &buffer[0], buffer.size(), 0, buffer.size() )

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>ByteBuffer</b> (p. 995).
---------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

6.169.3.40 **virtual ByteBuffer&** decaf::nio::ByteBuffer::putChar ( char *value* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** ) [pure virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

#### Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 978).



6.169.3.41 virtual **ByteBuffer&** decaf::nio::ByteBuffer::putChar ( int *index*, char *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes one byte containing the given value, into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 977).

6.169.3.42 virtual **ByteBuffer&** decaf::nio::ByteBuffer::putDouble ( double *value* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** ) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

#### Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 979).

6.169.3.43 `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble ( int index, double value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 978).

6.169.3.44 `virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat ( float value ) throw ( BufferOverflowException, ReadOnlyBufferException ) [pure virtual]`

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

#### Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 980).

6.169.3.45 virtual **ByteBuffer&** decaf::nio::ByteBuffer::putFloat ( int *index*, float *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 979).

6.169.3.46 virtual **ByteBuffer&** decaf::nio::ByteBuffer::putInt ( int *index*, int *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 980).

6.169.3.47 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt ( int value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

#### Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 981).

6.169.3.48 `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong ( long long value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

#### Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 981).

6.169.3.49 virtual **ByteBuffer&** decaf::nio::ByteBuffer::putLong ( int *index*, long long *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The value to write.

#### Returns

a reference to this buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 982).

6.169.3.50 virtual **ByteBuffer&** decaf::nio::ByteBuffer::putShort ( int *index*, short *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]

Writes two bytes containing the given value, into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The value to write.

#### Returns

a reference to this buffer

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 983).

6.169.3.51 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort ( short value ) throw ( BufferOverflowException, ReadOnlyBufferException )` [pure virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

#### Parameters

<i>value</i>	The value to be written.
--------------	--------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 914)	if there are fewer than bytes remaining in this buffer than the size of the data to be written.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 982).

6.169.3.52 `virtual ByteBuffer* decaf::nio::ByteBuffer::slice ( ) const` [pure virtual]

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **ByteBuffer** (p. 995) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 984).

6.169.3.53 `virtual std::string decaf::nio::ByteBuffer::toString ( ) const` [virtual]

#### Returns

a std::string describing this object

6.169.3.54 `static ByteBuffer* decaf::nio::ByteBuffer::wrap ( unsigned char * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new **ByteBuffer** (p. 995).

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the provided array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **ByteBuffer** (p. 995) that is backed by buffer, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array passed in is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.169.3.55 `static ByteBuffer* decaf::nio::ByteBuffer::wrap ( std::vector< unsigned char > & buffer ) [static]`

Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 995).

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new **ByteBuffer** (p. 995) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ByteBuffer.h`

## 6.170 cms::BytesMessage Class Reference

A **BytesMessage** (p. 1023) object is used to send a message containing a stream of unsigned bytes.

```
#include <src/main/cms/BytesMessage.h>
```

Inheritance diagram for cms::BytesMessage:

### Public Member Functions

- virtual **~BytesMessage** ()
- virtual void **setBodyBytes** (const unsigned char \*buffer, int numBytes)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*sets the bytes given to the message body.*
- virtual unsigned char \* **getBodyBytes** () const =0 throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.*
- virtual int **getBodyLength** () const =0 throw ( cms::MessageNotReadableException, cms::CMSEException )  
*Returns the number of bytes contained in the body of this message.*
- virtual void **reset** ()=0 throw ( cms::MessageFormatException, cms::CMSEException )  
*Puts the message body in read-only mode and repositions the stream of bytes to the beginning.*
- virtual bool **readBoolean** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Boolean from the Bytes message stream.*
- virtual void **writeBoolean** (bool value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a boolean to the bytes message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a Byte from the Bytes message stream.*
- virtual void **writeByte** (unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a byte to the bytes message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a byte array from the bytes message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )



*Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.*

- virtual int **readBytes** (unsigned char \*buffer, int length) const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a portion of the bytes message stream.*

- virtual void **writeBytes** (const unsigned char \*value, int offset, int length)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a portion of a byte array to the bytes message stream.*

- virtual char **readChar** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a Char from the Bytes message stream.*

- virtual void **writeChar** (char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a char to the bytes message stream as a 1-byte value.*

- virtual float **readFloat** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit float from the Bytes message stream.*

- virtual void **writeFloat** (float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a float to the bytes message stream as a 4 byte value.*

- virtual double **readDouble** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit double from the Bytes message stream.*

- virtual void **writeDouble** (double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a double to the bytes message stream as a 8 byte value.*

- virtual short **readShort** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit signed short from the Bytes message stream.*

- virtual void **writeShort** (short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed short to the bytes message stream as a 2 byte value.*

- virtual unsigned short **readUnsignedShort** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 16 bit unsigned short from the Bytes message stream.*

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a unsigned short to the bytes message stream as a 2 byte value.*

- virtual int **readInt** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 32 bit signed integer from the Bytes message stream.*

- virtual void **writeInt** (int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Writes a signed int to the bytes message stream as a 4 byte value.*

- virtual long long **readLong** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException )

*Reads a 64 bit long from the Bytes message stream.*

- virtual void **writeLong** (long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )

*Writes a long long to the bytes message stream as a 8 byte value.*

- virtual std::string **readString** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException )

*Reads an ASCII String from the Bytes message stream.*

- virtual void **writeString** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )

*Writes an ASCII String to the Bytes message stream.*

- virtual std::string **readUTF** () const =0 throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException )

*Reads an UTF String from the **BytesMessage** (p. 1023) stream.*

- virtual void **writeUTF** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )

*Writes an UTF String to the **BytesMessage** (p. 1023) stream.*

- virtual **BytesMessage** \* **clone** () const =0

*Clones this message.*

### 6.170.1 Detailed Description

A **BytesMessage** (p. 1023) object is used to send a message containing a stream of unsigned bytes.

It inherits from the **Message** (p. 2493) interface and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 1023) interface.

The **BytesMessage** (p. 1023) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1532) and **decaf.io.DataOutputStream** (p. 1546).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when **clearBody** is called, the body of the message is in write-only mode. After the first call to **reset** has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called **reset** so that the message body is in read-only mode for the client.

If **clearBody** is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 2679) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 2680) is thrown.

#### Since

1.0

### 6.170.2 Constructor & Destructor Documentation

6.170.2.1 `virtual cms::BytesMessage::~BytesMessage ( ) [inline, virtual]`

### 6.170.3 Member Function Documentation

6.170.3.1 `virtual BytesMessage* cms::BytesMessage::clone ( ) const [pure virtual]`

Clones this message.

#### Returns

a deep copy of this message.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs while cloning the <b>Message</b> (p. 2493).
-----------------------------------	---

Implements **cms::Message** (p. 2498).

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 205).

6.170.3.2 `virtual unsigned char* cms::BytesMessage::getBodyBytes ( ) const throw ( cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.

This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

#### Returns

pointer to a byte buffer that the call owns upon completion of this method.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

<b><i>MessageNotReadableException</i></b> (p. 2679)	- If the message is in Write Only Mode.
--	---

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 206).

6.170.3.3 `virtual int cms::BytesMessage::getBodyLength ( ) const throw ( cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Returns the number of bytes contained in the body of this message.

#### Returns

number of bytes.

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- If an internal error occurs.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 206).

6.170.3.4 `virtual bool cms::BytesMessage::readBoolean ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a Boolean from the Bytes message stream.

#### Returns

boolean value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 207).

6.170.3.5 `virtual unsigned char cms::BytesMessage::readByte ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a Byte from the Bytes message stream.

#### Returns

unsigned char value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 208).

6.170.3.6 `virtual int cms::BytesMessage::readBytes ( std::vector< unsigned char > & value ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

#### Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
---	---

<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 208).

```
6.170.3.7 virtual int cms::BytesMessage::readBytes ( unsigned char * buffer,
int length ) const throw ( cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException ) [pure
virtual]
```

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

#### Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 209).

6.170.3.8 `virtual char cms::BytesMessage::readChar ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a Char from the Bytes message stream.

#### Returns

char value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 210).

6.170.3.9 `virtual double cms::BytesMessage::readDouble ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]`

Reads a 64 bit double from the Bytes message stream.

#### Returns

double value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 210).

6.170.3.10 `virtual float cms::BytesMessage::readFloat ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 32 bit float from the Bytes message stream.

#### Returns

double value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p.210).

6.170.3.11 `virtual int cms::BytesMessage::readInt ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

#### Returns

int value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p.211).



6.170.3.12 `virtual long long cms::BytesMessage::readLong ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 64 bit long from the Bytes message stream.

#### Returns

long long value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 211).

6.170.3.13 `virtual short cms::BytesMessage::readShort ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 16 bit signed short from the Bytes message stream.

#### Returns

short value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 212).

6.170.3.14 `virtual std::string cms::BytesMessage::readString ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads an ASCII String from the Bytes message stream.

#### Returns

String from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 212).

6.170.3.15 `virtual unsigned short cms::BytesMessage::readUnsignedShort ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads a 16 bit unsigned short from the Bytes message stream.

#### Returns

unsigned short value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p. 213).

6.170.3.16 `virtual std::string cms::BytesMessage::readUTF ( ) const throw ( cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]`

Reads an UTF String from the **BytesMessage** (p. 1023) stream.

#### Returns

String from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 213).

6.170.3.17 `virtual void cms::BytesMessage::reset ( ) throw ( cms::MessageFormatException, cms::CMSEException ) [pure virtual]`

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- If the provider fails to perform the reset operation.
<b><i>MessageFormatException</i></b> (p. 2622)	- If the <b>Message</b> (p. 2493) has an invalid format.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 214).

6.170.3.18 `virtual void cms::BytesMessage::setBodyBytes ( const unsigned char * buffer, int numBytes ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

sets the bytes given to the message body.

#### Parameters

<i>buffer</i>	Byte Buffer to copy
<i>numBytes</i>	Number of bytes in Buffer to copy

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- If an internal error occurs.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if in Read Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 214).

```
6.170.3.19 virtual void cms::BytesMessage::writeBoolean ( bool value ) throw (
    cms::MessageNotWriteableException, cms::CMSException ) [pure
    virtual]
```

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

**Parameters**

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 215).

```
6.170.3.20 virtual void cms::BytesMessage::writeByte ( unsigned char value ) throw (
    cms::MessageNotWriteableException, cms::CMSException ) [pure
    virtual]
```

Writes a byte to the bytes message stream as a 1-byte value.

**Parameters**

<i>value</i>	byte to write to the stream
--------------	-----------------------------

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 215).

6.170.3.21 `virtual void cms::BytesMessage::writeBytes ( const std::vector< unsigned char > & value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
--------------	------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 216).

6.170.3.22 `virtual void cms::BytesMessage::writeBytes ( const unsigned char * value, int offset, int length ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a portion of a byte array to the bytes message stream.  
size as the number of bytes to write.

#### Parameters

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 215).

6.170.3.23 `virtual void cms::BytesMessage::writeChar ( char value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a char to the bytes message stream as a 1-byte value.

#### Parameters

<i>value</i>	char to write to the stream
--------------	-----------------------------

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b>MessageNotWriteableException</b> (p. 2680)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p.216).

6.170.3.24 `virtual void cms::BytesMessage::writeDouble ( double value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a double to the bytes message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b>MessageNotWriteableException</b> (p. 2680)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p.217).

6.170.3.25 `virtual void cms::BytesMessage::writeFloat ( float value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a float to the bytes message stream as a 4 byte value.

#### Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 217).

6.170.3.26 `virtual void cms::BytesMessage::writeInt ( int value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

**Parameters**

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 218).

6.170.3.27 `virtual void cms::BytesMessage::writeLong ( long long value ) throw ( cms::MessageNotWriteableException, cms::CMSEException )` [pure virtual]

Writes a long long to the bytes message stream as a 8 byte value.

**Parameters**

<i>value</i>	signed long long to write to the stream
--------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 218).

6.170.3.28 `virtual void cms::BytesMessage::writeShort ( short value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

#### Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p.218).

6.170.3.29 `virtual void cms::BytesMessage::writeString ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes an ASCII String to the Bytes message stream.

#### Parameters

<i>value</i>	String to write to the stream
--------------	-------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQBytesMessage` (p.219).

6.170.3.30 `virtual void cms::BytesMessage::writeUnsignedShort ( unsigned short value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

#### Parameters

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------



**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 219).

6.170.3.31 `virtual void cms::BytesMessage::writeUTF ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Writes an UTF String to the **BytesMessage** (p. 1023) stream.

**Parameters**

<i>value</i>	String to write to the stream
--------------	-------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of bytes stream has been reached.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 220).

The documentation for this class was generated from the following file:

- src/main/cms/**BytesMessage.h**

**6.171 activemq::cmsutil::CachedConsumer Class Reference**

A cached message consumer contained within a pooled session.

```
#include <src/main/activemq/cmsutil/CachedConsumer.h>
```

Inheritance diagram for activemq::cmsutil::CachedConsumer:

## Public Member Functions

- **CachedConsumer** (**cms::MessageConsumer** \*consumer)
- virtual **~CachedConsumer** ()
- virtual void **close** () throw ( cms::CMSEException )  
*Does nothing - the real producer resource will be closed by the lifecycle manager.*
- virtual **cms::Message** \* **receive** () throw ( cms::CMSEException )  
*Synchronously Receive a Message.*
- virtual **cms::Message** \* **receive** (int millisecs) throw ( cms::CMSEException )  
*Synchronously Receive a Message, time out after defined interval.*
- virtual **cms::Message** \* **receiveNoWait** () throw ( cms::CMSEException )  
*Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (**cms::MessageListener** \*listener) throw ( cms::CMSEException )  
*Sets the MessageListener that this class will send notifis on.*
- virtual **cms::MessageListener** \* **getMessageListener** () const throw ( cms::CMSEException )  
*Gets the MessageListener that this class will send mew Message notification events to.*
- virtual std::string **getMessageSelector** () const throw ( cms::CMSEException )  
*Gets this message consumer's message selector expression.*

## Protected Member Functions

- **CachedConsumer** (const **CachedConsumer** &)
- **CachedConsumer** & **operator=** (const **CachedConsumer** &)

### 6.171.1 Detailed Description

A cached message consumer contained within a pooled session.

### 6.171.2 Constructor & Destructor Documentation

- 6.171.2.1 **activemq::cmsutil::CachedConsumer::CachedConsumer** ( const **CachedConsumer** & ) [*inline, protected*]
- 6.171.2.2 **activemq::cmsutil::CachedConsumer::CachedConsumer** ( **cms::MessageConsumer** \* consumer ) [*inline*]
- 6.171.2.3 **virtual activemq::cmsutil::CachedConsumer::~~CachedConsumer** ( ) [*inline, virtual*]

### 6.171.3 Member Function Documentation

6.171.3.1 virtual void activemq::cmsutil::CachedConsumer::close ( ) throw ( cms::CMSEException ) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements cms::Closeable (p. 1120).

6.171.3.2 virtual cms::MessageListener\* activemq::cmsutil::CachedConsumer::getMessageListener ( ) const throw ( cms::CMSEException ) [inline, virtual]

Gets the MessageListener that this class will send new Message notification events to.

#### Returns

The listener of messages received by this consumer

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements cms::MessageConsumer (p. 2552).

6.171.3.3 virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector ( ) const throw ( cms::CMSEException ) [inline, virtual]

Gets this message consumer's message selector expression.

#### Returns

This Consumer's selector expression or "".

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements cms::MessageConsumer (p. 2552).

6.171.3.4 CachedConsumer& activemq::cmsutil::CachedConsumer::operator= ( const CachedConsumer & ) [inline, protected]

6.171.3.5 virtual cms::Message\* activemq::cmsutil::CachedConsumer::receive ( ) throw ( cms::CMSEException ) [inline, virtual]

Synchronously Receive a Message.

#### Returns

new message which the caller owns and must delete.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2553).

6.171.3.6 **virtual cms::Message\* activemq::cmsutil::CachedConsumer::receive ( int *millisecs* ) throw ( cms::CMSEException )** [inline, virtual]

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

**Returns**

new message which the caller owns and must delete.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2552).

6.171.3.7 **virtual cms::Message\* activemq::cmsutil::CachedConsumer::receiveNoWait ( )** throw ( cms::CMSEException ) [inline, virtual]

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

**Returns**

new message which the caller owns and must delete.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2553).

6.171.3.8 **virtual void activemq::cmsutil::CachedConsumer::setMessageListener ( cms::MessageListener \* *listener* ) throw ( cms::CMSEException )** [inline, virtual]

Sets the MessageListener that this class will send notifs on.

**Parameters**

<i>listener</i>	The listener of messages received by this consumer.
-----------------	---

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageConsumer** (p. 2553).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedConsumer.h**

**6.172 activemq::cmsutil::CachedProducer Class Reference**

A cached message producer contained within a pooled session.

```
#include <src/main/activemq/cmsutil/CachedProducer.h>
```

Inheritance diagram for activemq::cmsutil::CachedProducer:

**Public Member Functions**

- **CachedProducer** (**cms::MessageProducer** \*producer)
- virtual **~CachedProducer** ()
- virtual void **close** () throw ( cms::CMSEException )  
*Does nothing - the real producer resource will be closed by the lifecycle manager.*
- virtual void **send** (**cms::Message** \*message) throw ( cms::CMSEException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (**cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message) throw ( cms::CMSEException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **cms::Destination** \*destination, **cms::Message** \*message, int deliveryMode, int priority, long long timeToLive) throw ( cms::CMSEException )  
*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **setDeliveryMode** (int mode) throw ( cms::CMSEException )  
*Sets the delivery mode for this Producer.*
- virtual int **getDeliveryMode** () const throw ( cms::CMSEException )  
*Gets the delivery mode for this Producer.*
- virtual void **setDisableMessageID** (bool value) throw ( cms::CMSEException )  
*Sets if Message Ids are disabled for this Producer.*

- virtual bool **getDisableMessageID** () const throw ( cms::CMSEException )  
*Gets if Message Ids are disabled for this Producer.*
- virtual void **setDisableMessageTimeStamp** (bool value) throw ( cms::CMSEException )  
*Sets if Message Time Stamps are disabled for this Producer.*
- virtual bool **getDisableMessageTimeStamp** () const throw ( cms::CMSEException )  
*Gets if Message Time Stamps are disabled for this Producer.*
- virtual void **setPriority** (int priority) throw ( cms::CMSEException )  
*Sets the Priority that this Producers sends messages at.*
- virtual int **getPriority** () const throw ( cms::CMSEException )  
*Gets the Priority level that this producer sends messages at.*
- virtual void **setTimeToLive** (long long time) throw ( cms::CMSEException )  
*Sets the Time to Live that this Producers sends messages with.*
- virtual long long **getTimeToLive** () const throw ( cms::CMSEException )  
*Gets the Time to Live that this producer sends messages with.*

### Protected Member Functions

- **CachedProducer** (const **CachedProducer** &)
- **CachedProducer** & **operator=** (const **CachedProducer** &)

#### 6.172.1 Detailed Description

A cached message producer contained within a pooled session.

#### 6.172.2 Constructor & Destructor Documentation

- 6.172.2.1 **activemq::cmsutil::CachedProducer::CachedProducer** ( const **CachedProducer** & ) [inline, protected]
- 6.172.2.2 **activemq::cmsutil::CachedProducer::CachedProducer** ( cms::MessageProducer \* *producer* ) [inline]
- 6.172.2.3 **virtual activemq::cmsutil::CachedProducer::~~CachedProducer** ( ) [inline, virtual]

#### 6.172.3 Member Function Documentation

- 6.172.3.1 **virtual void activemq::cmsutil::CachedProducer::close** ( ) throw ( cms::CMSEException ) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 1120).

6.172.3.2 `virtual int activemq::cmsutil::CachedProducer::getDeliveryMode ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the delivery mode for this Producer.

#### Returns

The DeliveryMode

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2683).

6.172.3.3 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

#### Returns

boolean indicating enable / disable (true / false)

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2683).

6.172.3.4 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

#### Returns

boolean indicating enable / disable (true / false)

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2684).

6.172.3.5 `virtual int activemq::cmsutil::CachedProducer::getPriority ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

**Returns**

int based priority level

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2684).

6.172.3.6 `virtual long long activemq::cmsutil::CachedProducer::getTimeToLive ( ) const throw ( cms::CMSEException ) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

**Returns**

Time to live value in milliseconds

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2684).

6.172.3.7 `CachedProducer& activemq::cmsutil::CachedProducer::operator= ( const CachedProducer & ) [inline, protected]`

6.172.3.8 `virtual void activemq::cmsutil::CachedProducer::send ( cms::Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSEException ) [inline, virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

**Parameters**

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

**Exceptions**

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.



<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.
--------------------------------------	--

Implements **cms::MessageProducer** (p. 2685).

**6.172.3.9** `virtual void activemq::cmsutil::CachedProducer::send ( const cms::Destination * destination, cms::Message * message ) throw ( cms::CMSExcption ) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

#### Exceptions

<i>CMSExcption</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperationException</i>	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2687).

**6.172.3.10** `virtual void activemq::cmsutil::CachedProducer::send ( cms::Message * message ) throw ( cms::CMSExcption ) [inline, virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

#### Exceptions

<i>CMSExcption</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.

<i>UnsupportedOperation</i> Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.
---------------------------------------	--

Implements **cms::MessageProducer** (p. 2686).

6.172.3.11 `virtual void activemq::cmsutil::CachedProducer::send ( const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSEException ) [inline, virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs while sending the message.
<i>MessageFormatException</i>	- if an Invalid Message is given.
<i>InvalidDestinationException</i>	- if a client uses this method with a MessageProducer with an invalid destination.
<i>UnsupportedOperation</i> Exception	- if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2685).

6.172.3.12 `virtual void activemq::cmsutil::CachedProducer::setDeliveryMode ( int mode ) throw ( cms::CMSEException ) [inline, virtual]`

Sets the delivery mode for this Producer.

#### Parameters

<i>mode</i>	The DeliveryMode
-------------	------------------

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2687).

6.172.3.13 virtual void activemq::cmsutil::CachedProducer::setDisableMessageID ( bool *value* )  
throw ( cms::CMSEException ) [inline, virtual]

Sets if Message Ids are disabled for this Producer.

#### Parameters

<i>value</i>	boolean indicating enable / disable (true / false)
--------------	--

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p.2688).

6.172.3.14 virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp ( bool *value* ) throw ( cms::CMSEException ) [inline, virtual]

Sets if Message Time Stamps are disabled for this Producer.

#### Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p.2688).

6.172.3.15 virtual void activemq::cmsutil::CachedProducer::setPriority ( int *priority* ) throw ( cms::CMSEException ) [inline, virtual]

Sets the Priority that this Producers sends messages at.

#### Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

#### Exceptions

<i>CMSEException</i>	- if an internal error occurs.
----------------------	--------------------------------

Implements **cms::MessageProducer** (p.2688).

6.172.3.16 virtual void activemq::cmsutil::CachedProducer::setTimeToLive ( long long *time* )  
throw ( cms::CMSEException ) [inline, virtual]

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

#### Parameters

<i>time</i>	default time to live value in milliseconds
-------------	--

#### Exceptions

<i>CMSException</i>	- if an internal error occurs.
---------------------	--------------------------------

Implements **cms::MessageProducer** (p. 2689).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedProducer.h**

## 6.173 decaf::util::concurrent::Callable< V > Class Template Reference

A task that returns a result and may throw an exception.

```
#include <src/main/decaf/util/concurrent/Callable.h>
```

### Public Member Functions

- virtual **~Callable** ()
- virtual **V call** ()=0 throw ( decaf::lang::Exception )  
*Computes a result, or throws an exception if unable to do so.*

#### 6.173.1 Detailed Description

```
template<typename V>class decaf::util::concurrent::Callable< V >
```

A task that returns a result and may throw an exception.

Implementors define a single method with no arguments called call. This interface differs from the Runnable interface in that a **Callable** (p. 1051) object can return a result and is allowed to throw an exceptions from its call method.

The Executors class contains utility methods to convert from other common forms to **Callable** (p. 1051) classes.

#### Since

1.0

## 6.173.2 Constructor & Destructor Documentation

6.173.2.1 `template<typename V > virtual decaf::util::concurrent::Callable< V >::~~Callable ( ) [inline, virtual]`

## 6.173.3 Member Function Documentation

6.173.3.1 `template<typename V > virtual V decaf::util::concurrent::Callable< V >::call ( ) throw ( decaf::lang::Exception ) [pure virtual]`

Computes a result, or throws an exception if unable to do so.

### Returns

Computed Result.

### Exceptions

<i>Exception</i>	If unable to compute a result.
------------------	--------------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Callable.h`

## 6.174 decaf::util::concurrent::CancellationException Class Reference

```
#include <src/main/decaf/util/concurrent/CancellationException.h>
```

Inheritance diagram for `decaf::util::concurrent::CancellationException`:

### Public Member Functions

- **CancellationException** ( ) throw ( )  
*Default Constructor.*
- **CancellationException** (const **decaf::lang::Exception** &ex) throw ( )  
*Conversion Constructor from some other Exception.*
- **CancellationException** (const **CancellationException** &ex) throw ( )  
*Copy Constructor.*
- **CancellationException** (const std::exception \*cause) throw ( )  
*Constructor.*
- **CancellationException** (const char \*file, const int lineNumber, const char \*msg,...) throw ( )  
*Constructor - Initializes the file name and line number where this message occurred.*

- **CancellationException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CancellationException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CancellationException** () throw ()

### 6.174.1 Constructor & Destructor Documentation

6.174.1.1 **decaf::util::concurrent::CancellationException::CancellationException** ( ) throw ()  
[inline]

Default Constructor.

6.174.1.2 **decaf::util::concurrent::CancellationException::CancellationException** ( const **decaf::lang::Exception** & **ex** ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<b>ex</b>	An exception that should become this type of Exception
-----------	--

6.174.1.3 **decaf::util::concurrent::CancellationException::CancellationException** ( const **CancellationException** & **ex** ) throw () [inline]

Copy Constructor.

#### Parameters

<b>ex</b>	- The Exception to copy in this new instance.
-----------	---

6.174.1.4 **decaf::util::concurrent::CancellationException::CancellationException** ( const std::exception \* **cause** ) throw () [inline]

Constructor.

#### Parameters

<b>cause</b>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.174.1.5 `decaf::util::concurrent::CancellationException::CancellationException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.174.1.6 `decaf::util::concurrent::CancellationException::CancellationException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.174.1.7 `virtual decaf::util::concurrent::CancellationException::~~CancellationException ( ) throw () [inline, virtual]`

## 6.174.2 Member Function Documentation

6.174.2.1 `virtual CancellationException* decaf::util::concurrent::CancellationException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CancellationException.h**

## 6.175 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/Certificate.h>
```

Inheritance diagram for decaf::security::cert::Certificate:

### Public Member Functions

- virtual **~Certificate** ()
- virtual bool **equals** (const **Certificate** &cert) const =0  
*Compares the encoded form of the two certificates.*
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw ( CertificateEncodingException )  
*Provides the encoded form of this certificate.*
- virtual std::string **getType** () const =0  
*Returns the type of this certificate.*
- virtual **PublicKey** \* **getPublicKey** ()=0  
*Gets the public key of this certificate.*
- virtual const **PublicKey** \* **getPublicKey** () const =0  
*Gets the public key of this certificate.*
- virtual void **verify** (const **PublicKey** &publicKey) const =0 throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)  
*Verifies that this certificate was signed with the private key that corresponds to the specified public key.*
- virtual void **verify** (const **PublicKey** &publicKey, const std::string &sigProvider) const =0 throw ( NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)  
*Verifies that this certificate was signed with the private key that corresponds to the specified public key.*
- virtual std::string **toString** () const =0  
*Returns a string representation of this certificate.*

### 6.175.1 Detailed Description

Base interface for all identity certificates.



## 6.175.2 Constructor & Destructor Documentation

6.175.2.1 `virtual decaf::security::cert::Certificate::~~Certificate ( ) [inline, virtual]`

## 6.175.3 Member Function Documentation

6.175.3.1 `virtual bool decaf::security::cert::Certificate::equals ( const Certificate & cert ) const [pure virtual]`

Compares the encoded form of the two certificates.

### Parameters

<i>cert</i>	The certificate to be tested for equality with this certificate.
-------------	--

### Returns

true if the given certificate is equal to this certificate.

6.175.3.2 `virtual void decaf::security::cert::Certificate::getEncoded ( std::vector< unsigned char > & output ) const throw ( CertificateEncodingException ) [pure virtual]`

Provides the encoded form of this certificate.

### Parameters

<i>output</i>	Receives the encoded form of this certificate.
---------------	--

### Exceptions

<b><i>CertificateEncodingException</i></b> (p. 1059)	if an encoding error occurs
---	-----------------------------

6.175.3.3 `virtual PublicKey* decaf::security::cert::Certificate::getPublicKey ( ) [pure virtual]`

Gets the public key of this certificate.

### Returns

the public key

6.175.3.4 `virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey ( ) const`  
[pure virtual]

Gets the public key of this certificate.

#### Returns

the public key

6.175.3.5 `virtual std::string decaf::security::cert::Certificate::getType ( ) const` [pure  
virtual]

Returns the type of this certificate.

#### Returns

the type of this certificate

6.175.3.6 `virtual std::string decaf::security::cert::Certificate::toString ( ) const` [pure  
virtual]

Returns a string representation of this certificate.

#### Returns

a string representation of this certificate

6.175.3.7 `virtual void decaf::security::cert::Certificate::verify ( const PublicKey & publicKey,  
const std::string & sigProvider ) const throw ( NoSuchAlgorithmException,  
InvalidKeyException, NoSuchProviderException, SignatureException,  
CertificateException)` [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the  
specified public key.

Uses the verification engine of the specified provider.

#### Parameters

<i>publicKey</i>	The public key used to carry out the validation.
<i>sigProvider</i>	The name of the signature provider

#### Exceptions

<b>NoSuchAlgorithmException</b> (p. 2776)	- on unsupported signature algorithms.
--	--

<b><i>InvalidKeyException</i></b> (p. 2094)	- on incorrect key.
<b><i>NoSuchProviderException</i></b> (p. 2781)	- if there's no default provider.
<b><i>SignatureException</i></b> (p. 3440)	- on signature errors.
<b><i>CertificateException</i></b> (p. 1061)	- on encoding errors.

6.175.3.8 virtual void decaf::security::cert::Certificate::verify ( const PublicKey & *publicKey*  
) const throw ( NoSuchAlgorithmException, InvalidKeyException,  
NoSuchProviderException, SignatureException, CertificateException)  
[pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

#### Parameters

<i>publicKey</i>	The public key used to carry out the validation.
------------------	--

#### Exceptions

<b><i>NoSuchAlgorithmException</i></b> (p. 2776)	- on unsupported signature algorithms.
<b><i>InvalidKeyException</i></b> (p. 2094)	- on incorrect key.
<b><i>NoSuchProviderException</i></b> (p. 2781)	- if there's no default provider.
<b><i>SignatureException</i></b> (p. 3440)	- on signature errors.
<b><i>CertificateException</i></b> (p. 1061)	- on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**

## 6.176 decaf::security::cert::CertificateEncodingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateEncodingException.h>
```

Inheritance diagram for decaf::security::cert::CertificateEncodingException:

### Public Member Functions

- **CertificateEncodingException** () throw ()  
*Default Constructor.*
- **CertificateEncodingException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()  
*Copy Constructor.*
- **CertificateEncodingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateEncodingException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateEncodingException** () throw ()

### 6.176.1 Constructor & Destructor Documentation

6.176.1.1 decaf::security::cert::CertificateEncodingException::CertificateEncodingException ( ) throw () [inline]

Default Constructor.

6.176.1.2 decaf::security::cert::CertificateEncodingException::CertificateEncodingException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<b>ex</b>	An exception that should become this type of Exception
-----------	--

## 6.176 decaf::security::cert::CertificateEncodingException Class Reference 1063

6.176.1.3 decaf::security::cert::CertificateEncodingException::CertificateEncodingException ( const CertificateEncodingException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.176.1.4 decaf::security::cert::CertificateEncodingException::CertificateEncodingException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.176.1.5 virtual decaf::security::cert::CertificateEncodingException::~~CertificateEncodingException ( ) throw () [inline, virtual]

## 6.176.2 Member Function Documentation

6.176.2.1 virtual CertificateEncodingException\* decaf::security::cert::CertificateEncodingException::clone ( ) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 1062).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateEncodingException.h**

## 6.177 decaf::security::cert::CertificateException Class Reference

```
#include <src/main/decaf/security/cert/CertificateException.h>
```

Inheritance diagram for decaf::security::cert::CertificateException:

### Public Member Functions

- **CertificateException** () throw ()  
*Default Constructor.*
- **CertificateException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateException** (const **CertificateException** &ex) throw ()  
*Copy Constructor.*
- **CertificateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateException** () throw ()

### 6.177.1 Constructor & Destructor Documentation

6.177.1.1 decaf::security::cert::CertificateException::CertificateException ( ) throw ()  
[inline]

Default Constructor.

6.177.1.2 decaf::security::cert::CertificateException::CertificateException ( const Exception &ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

ex	An exception that should become this type of Exception
----	--

6.177.1.3 decaf::security::cert::CertificateException::CertificateException ( const CertificateException &ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

**6.177.1.4** `decaf::security::cert::CertificateException::CertificateException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

**6.177.1.5** `virtual decaf::security::cert::CertificateException::~CertificateException ( ) throw ()  
[inline, virtual]`

## 6.177.2 Member Function Documentation

**6.177.2.1** `virtual CertificateException* decaf::security::cert::CertificateException::clone ( )  
const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1936).

Reimplemented in **decaf::security::cert::CertificateEncodingException** (p. 1061), **decaf::security::cert::CertificateExpiredException** (p. 1064), **decaf::security::cert::CertificateNotYetValidException** (p. 1066), and **decaf::security::cert::CertificateParsingException** (p. 1068).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateException.h`

## 6.178 decaf::security::cert::CertificateExpiredException Class Reference

```
#include <src/main/decaf/security/cert/CertificateExpiredException.h>
```

Inheritance diagram for decaf::security::cert::CertificateExpiredException:

### Public Member Functions

- **CertificateExpiredException** () throw ()  
*Default Constructor.*
- **CertificateExpiredException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateExpiredException** (const **CertificateExpiredException** &ex) throw ()  
*Copy Constructor.*
- **CertificateExpiredException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateExpiredException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateExpiredException** () throw ()

### 6.178.1 Constructor & Destructor Documentation

6.178.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( ) throw () [inline]

Default Constructor.

6.178.1.2 decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

ex	An exception that should become this type of Exception
----	--



6.178.1.3 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( const CertificateExpiredException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.178.1.4 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.178.1.5 `virtual decaf::security::cert::CertificateExpiredException::~~CertificateExpiredException ( ) throw () [inline, virtual]`

## 6.178.2 Member Function Documentation

6.178.2.1 `virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 1062).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateExpiredException.h`

## 6.179 decaf::security::cert::CertificateNotYetValidException Class Reference

```
#include <src/main/decaf/security/cert/CertificateNotYetValidException.h>
```

Inheritance diagram for decaf::security::cert::CertificateNotYetValidException:

### Public Member Functions

- **CertificateNotYetValidException** () throw ()  
*Default Constructor.*
- **CertificateNotYetValidException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateNotYetValidException** (const **CertificateNotYetValidException** &ex) throw ()  
*Copy Constructor.*
- **CertificateNotYetValidException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateNotYetValidException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateNotYetValidException** () throw ()

### 6.179.1 Constructor & Destructor Documentation

6.179.1.1 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException ( ) throw () [inline]

Default Constructor.

6.179.1.2 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

### Parameters

ex	An exception that should become this type of Exception
----	--

## 6.179 decaf::security::cert::CertificateNotYetValidException Class Reference 1069

6.179.1.3 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException  
( const CertificateNotYetValidException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.179.1.4 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException  
( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.179.1.5 virtual decaf::security::cert::CertificateNotYetValidException::~~CertificateNotYetValidException  
( ) throw () [inline, virtual]

## 6.179.2 Member Function Documentation

6.179.2.1 virtual CertificateNotYetValidException\*  
decaf::security::cert::CertificateNotYetValidException::clone ( ) const [inline,  
virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 1062).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/CertificateNotYetValidException.h

## 6.180 decaf::security::cert::CertificateParsingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateParsingException.h>
```

Inheritance diagram for decaf::security::cert::CertificateParsingException:

### Public Member Functions

- **CertificateParsingException** () throw ()  
*Default Constructor.*
- **CertificateParsingException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()  
*Copy Constructor.*
- **CertificateParsingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **CertificateParsingException** \* **clone** () const  
*Clones this exception.*
- virtual ~**CertificateParsingException** () throw ()

### 6.180.1 Constructor & Destructor Documentation

6.180.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException ( )  
throw () [inline]

Default Constructor.

6.180.1.2 decaf::security::cert::CertificateParsingException::CertificateParsingException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

ex	An exception that should become this type of Exception
----	--

## 6.180 decaf::security::cert::CertificateParsingException Class Reference 1071

6.180.1.3 decaf::security::cert::CertificateParsingException::CertificateParsingException ( const CertificateParsingException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.180.1.4 decaf::security::cert::CertificateParsingException::CertificateParsingException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.180.1.5 virtual decaf::security::cert::CertificateParsingException::~~CertificateParsingException ( ) throw () [inline, virtual]

## 6.180.2 Member Function Documentation

6.180.2.1 virtual CertificateParsingException\* decaf::security::cert::CertificateParsingException::clone ( ) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 1062).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateParsingException.h**

## 6.181 decaf::lang::Character Class Reference

```
#include <src/main/decaf/lang/Character.h>
```

Inheritance diagram for decaf::lang::Character:

### Public Member Functions

- **Character** (char value)
- virtual int **compareTo** (const **Character** &c) const  
*Compares this **Character** (p. 1069) instance with another.*
- virtual bool **operator==** (const **Character** &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Character** &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const char &c) const  
*Compares this **Character** (p. 1069) instance with a char type.*
- virtual bool **operator==** (const char &c) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const char &c) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static **Character valueOf** (char value)  
*Returns a **Character** (p. 1069) instance representing the specified char value.*
- static bool **isWhitespace** (char c)  
*Indicates whether or not the given character is considered whitespace.*
- static bool **isDigit** (char c)  
*Indicates whether or not the given character is a digit.*
- static bool **isLowerCase** (char c)  
*Indicates whether or not the given character is a lower case character.*
- static bool **isUpperCase** (char c)  
*Indicates whether or not the given character is a upper case character.*
- static bool **isLetter** (char c)  
*Indicates whether or not the given character is a letter.*
- static bool **isLetterOrDigit** (char c)  
*Indicates whether or not the given character is either a letter or a digit.*
- static bool **isISOControl** (char c)  
*Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.*
- static int **digit** (char c, int radix)  
*Returns the numeric value of the character ch in the specified radix.*

## Static Public Attributes

- static const int **MIN\_RADIX** = 2  
*The minimum radix available for conversion to and from strings.*
- static const int **MAX\_RADIX** = 36  
*The maximum radix available for conversion to and from strings.*
- static const char **MIN\_VALUE** = (char)0x7F  
*The minimum value that a signed char can take on.*
- static const char **MAX\_VALUE** = (char)0x80  
*The maximum value that a signed char can take on.*
- static const int **SIZE** = 8  
*The size of the primitive charactor in bits.*

## 6.181.1 Constructor & Destructor Documentation

### 6.181.1.1 decaf::lang::Character::Character ( char value )

#### Parameters

<i>value</i>	- char to wrap.
--------------	-----------------

## 6.181.2 Member Function Documentation

6.181.2.1 `virtual unsigned char decaf::lang::Character::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2787).

6.181.2.2 `virtual int decaf::lang::Character::compareTo ( const char & c ) const [inline, virtual]`

Compares this **Character** (p. 1069) instance with a char type.

### Parameters

<code>c</code>	- the char instance to be compared
----------------	------------------------------------

### Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **char** > (p. 1187).

6.181.2.3 `virtual int decaf::lang::Character::compareTo ( const Character & c ) const [inline, virtual]`

Compares this **Character** (p. 1069) instance with another.

### Parameters

<code>c</code>	- the <b>Character</b> (p. 1069) instance to be compared
----------------	--

### Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Character** > (p. 1187).



**6.181.2.4** static int decaf::lang::Character::digit ( char *c*, int *radix* ) [static]

Returns the numeric value of the character *ch* in the specified radix.

If the radix is not in the range `MIN_RADIX <= radix <= MAX_RADIX` or if the value of *ch* is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

\* The method `isDigit` is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned. \* The character is one of the uppercase Latin letters 'A' through 'Z' and its code is less than `radix + 'A' - 10`. In this case, `ch - 'A' + 10` is returned. \* The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than `radix + 'a' - 10`. In this case, `ch - 'a' + 10` is returned.

**Parameters**

<i>c</i>	- the char to be converted
<i>radix</i>	- the radix of the number

**Returns**

the numeric value of the number represented in the given radix

**6.181.2.5** virtual double decaf::lang::Character::doubleValue ( ) const [inline, virtual]

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p.2787).

**6.181.2.6** bool decaf::lang::Character::equals ( const char & *c* ) const [inline, virtual]**Returns**

true if the two Characters have the same value.

Implements **decaf::lang::Comparable< char >** (p.1188).

**6.181.2.7** bool decaf::lang::Character::equals ( const Character & *c* ) const [inline, virtual]**Returns**

true if the two **Character** (p.1069) Objects have the same value.

Implements **decaf::lang::Comparable< Character >** (p.1188).

6.181.2.8 `virtual float decaf::lang::Character::floatValue ( ) const [inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.181.2.9 `virtual int decaf::lang::Character::intValue ( ) const [inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.181.2.10 `static bool decaf::lang::Character::isDigit ( char c ) [inline, static]`

Indicates whether or not the given character is a digit.

6.181.2.11 `static bool decaf::lang::Character::isISOControl ( char c ) [inline, static]`

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

#### Parameters

<code>c</code>	- the character, including supplementary characters
----------------	---

#### Returns

true if the char is an ISO control character

6.181.2.12 `static bool decaf::lang::Character::isLetter ( char c ) [inline, static]`

Indicates whether or not the given character is a letter.

6.181.2.13 `static bool decaf::lang::Character::isLetterOrDigit ( char c ) [inline, static]`

Indicates whether or not the given character is either a letter or a digit.

6.181.2.14 `static bool decaf::lang::Character::isLowerCase ( char c ) [inline, static]`

Indicates whether or not the given character is a lower case character.

6.181.2.15 `static bool decaf::lang::Character::isUpperCase ( char c ) [inline, static]`

Indicates whether or not the given character is a upper case character.

6.181.2.16 `static bool decaf::lang::Character::isWhitespace ( char c ) [inline, static]`

Indicates whether or not the given character is considered whitespace.

6.181.2.17 `virtual long long decaf::lang::Character::longValue ( ) const [inline, virtual]`

Answers the long value which the receiver represents.

### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.181.2.18 `virtual bool decaf::lang::Character::operator< ( const Character & c ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Character >** (p. 1188).

6.181.2.19 `virtual bool decaf::lang::Character::operator< ( const char & c ) const`  
`[inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **char** > (p. 1188).

6.181.2.20 `virtual bool decaf::lang::Character::operator== ( const Character & c ) const`  
`[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Character** > (p. 1189).

6.181.2.21 `virtual bool decaf::lang::Character::operator== ( const char & c ) const`  
`[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<code>c</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **char** > (p. 1189).

6.181.2.22 `virtual short decaf::lang::Character::shortValue ( ) const [inline, virtual]`

Answers the short value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2788).

6.181.2.23 `std::string decaf::lang::Character::toString ( ) const`

#### Returns

this **Character** (p. 1069) Object as a **String** (p. 3610) Representation

6.181.2.24 `static Character decaf::lang::Character::valueOf ( char value ) [inline, static]`

Returns a **Character** (p. 1069) instance representing the specified char value.

#### Parameters

<i>value</i>	- the primitive char to wrap.
--------------	-------------------------------

#### Returns

a new Charactor instance that wraps this value.

### 6.181.3 Field Documentation

6.181.3.1 `const int decaf::lang::Character::MAX_RADIX = 36 [static]`

The maximum radix available for conversion to and from strings.

6.181.3.2 `const char decaf::lang::Character::MAX_VALUE = (char)0x80 [static]`

The maximum value that a signed char can take on.

6.181.3.3 `const int decaf::lang::Character::MIN_RADIX = 2 [static]`

The minimum radix available for conversion to and from strings.

6.181.3.4 `const char decaf::lang::Character::MIN_VALUE = (char)0x7F` `[static]`

The minimum value that a signed char can take on.

6.181.3.5 `const int decaf::lang::Character::SIZE = 8` `[static]`

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Character.h`

## 6.182 decaf::internal::nio::CharArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/CharArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::CharArrayBuffer`:

### Public Member Functions

- **CharArrayBuffer** (int size, bool **readOnly**=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **CharArrayBuffer** (p. 1077) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **CharArrayBuffer** (char \*array, int size, int **offset**, int **length**, bool **readOnly**=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a **CharArrayBuffer** (p. 1077) object that wraps the given array.*
- **CharArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int **offset**, int **length**, bool **readOnly**=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.*
- **CharArrayBuffer** (const **CharArrayBuffer** &other)  
*Create a **CharArrayBuffer** (p. 1077) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.*
- virtual **~CharArrayBuffer** ()
- virtual char \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )  
*Returns the character array that backs this buffer (optional operation).  
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.  
Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.*

**Returns**

the array that backs this **Buffer** (p. 887).

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual CharBuffer \* **asReadOnlyBuffer** () const

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns**

The new, read-only char buffer which the caller then owns.

- virtual CharBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **CharBuffer** (p. 1089).

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3115)	- If this buffer is read-only
---	-------------------------------

- virtual CharBuffer \* **duplicate** ()

*Creates a new char buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*a new char **Buffer** (p. 887) which the caller owns.*

- virtual char **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the character at this buffer's current position, and then increments the position.*

**Returns**

*the char at the current position.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return
---	---------------------------------

- virtual char **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the char at the given index.*

**Parameters**

index	The index in the <b>Buffer</b> (p. 887) where the char is to be read.
-------	---

**Returns**

*the char that is located at the given index.*

**Exceptions**

IndexOutOfBoundsException	if index is not smaller than the buffer's limit or is negative.
---------------------------	---

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible char array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*



- virtual CharBuffer & **put** (char value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given char into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The char value to be written.</i>
-------	--------------------------------------

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	<i>if this buffer's current position is not smaller than its limit</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only.</i>

- virtual CharBuffer & **put** (int index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given char into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 887) to write the data.</i>
value	<i>The char to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or index is negative.</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only.</i>

- virtual CharBuffer \* **slice** () const

*Creates a new **CharBuffer** (p. 1089) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the newly create **CharBuffer** (p. 1089) which the caller owns.*

- virtual lang::CharSequence \* **subSequence** (int start, int end) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.*

*The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new*

buffer's capacity will be that of this buffer, its position will be **position()** (p. 892) + start, and its limit will be **position()** (p. 892) + end. The new **Buffer** (p. 887) will be read-only if, and only if, this buffer is read-only.

#### Parameters

start	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than <b>remaining()</b> (p. 892).
end	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than <b>remaining()</b> (p. 892).

#### Returns

The new character buffer, caller owns.

#### Exceptions

IndexOutOfBoundsException	if the preconditions on start and end fail.
---------------------------	---

### Protected Member Functions

- virtual void **setReadOnly** (bool value)  
Sets this **CharArrayBuffer** (p. 1077) as Read-Only.

### Protected Attributes

- bool **readOnly**
- **decaf::lang::Pointer**< **ByteArrayAdapter** > **\_array**
- int **offset**
- int **length**

### 6.182.1 Constructor & Destructor Documentation

6.182.1.1 **decaf::internal::nio::CharArrayBuffer::CharArrayBuffer** ( int size, bool readOnly = false ) throw ( **decaf::lang::exceptions::IllegalArgumentException** )

Creates a **CharArrayBuffer** (p. 1077) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<b>IllegalArgumentException</b>	if the capacity value is negative.
---------------------------------	------------------------------------

```
6.182.1.2 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer ( char *
array, int size, int offset, int length, bool readOnly = false )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException )
```

Creates a **CharArrayBuffer** (p. 1077) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

```
6.182.1.3 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer ( const
decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool
readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException )
```

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **CharArrayBuffer** (p. 1077) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.182.1.4 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer ( const CharArrayBuffer & other )`

Create a **CharArrayBuffer** (p. 1077) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>CharArrayBuffer</b> (p. 1077) this one is to mirror.
--------------	---

6.182.1.5 `virtual decaf::internal::nio::CharArrayBuffer::~~CharArrayBuffer ( ) [virtual]`

### 6.182.2 Member Function Documentation

6.182.2.1 `virtual char* decaf::internal::nio::CharArrayBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 887).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1095).

6.182.2.2 `virtual int decaf::internal::nio::CharArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 1095).

**6.182.2.3** `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::asReadOnlyBuffer ( ) const [virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns**

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 1096).

**6.182.2.4** `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException ) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **CharBuffer** (p. 1089).

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	- If this buffer is read-only
--	-------------------------------

Implements **decaf::nio::CharBuffer** (p. 1096).

6.182.2.5 **virtual CharBuffer\*** **decaf::internal::nio::CharArrayBuffer::duplicate ( )**  
[virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new char **Buffer** (p. 887) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1097).

6.182.2.6 **virtual char** **decaf::internal::nio::CharArrayBuffer::get ( )** throw (**decaf::nio::BufferUnderflowException**) [virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

**Returns**

the char at the current position.

**Exceptions**

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return
--	---------------------------------

Implements **decaf::nio::CharBuffer** (p. 1097).

6.182.2.7 **virtual char** **decaf::internal::nio::CharArrayBuffer::get ( int *index* )** const throw (**lang::exceptions::IndexOutOfBoundsException**) [virtual]

Absolute get method.

Reads the char at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the char is to be read.
--------------	---

#### Returns

the char that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit or is negative.
----------------------------------	---

Implements **decaf::nio::CharBuffer** (p. 1098).

6.182.2.8 `virtual bool decaf::internal::nio::CharArrayBuffer::hasArray ( ) const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 1099).

6.182.2.9 `virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly ( ) const [inline, virtual]`

Tells whether or not this buffer is read-only.

#### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 890).

6.182.2.10 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put ( char value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given char into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The char value to be written.
--------------	-------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not smaller than its limit
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1101).

6.182.2.11 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put ( int index, char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given char into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The char to write.

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>IndexOutOfBoundsException</i></b>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::CharBuffer** (p. 1102).

6.182.2.12 `virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly ( bool value )`  
[inline, protected, virtual]

Sets this **CharArrayBuffer** (p. 1077) as Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---



6.182.2.13 `virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **CharBuffer** (p. 1089) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **CharBuffer** (p. 1089) which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 1105).

6.182.2.14 `virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence`  
`( int start, int end ) const throw ( de-`  
`caf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 892) + start, and its limit will be **position()** (p. 892) + end. The new **Buffer** (p. 887) will be read-only if, and only if, this buffer is read-only.

### Parameters

<i>start</i>	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than <b>remaining()</b> (p. 892).
<i>end</i>	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than <b>remaining()</b> (p. 892).

### Returns

The new character buffer, caller owns.

### Exceptions

<i>IndexOutOfBoundsException</i>	if the preconditions on start and end fail.
----------------------------------	---

Implements **decaf::nio::CharBuffer** (p. 1105).

### 6.182.3 Field Documentation

- 6.182.3.1 **decaf::lang::Pointer<ByteArrayAdapter>**  
**decaf::internal::nio::CharArrayBuffer::\_array** [protected]
- 6.182.3.2 **int decaf::internal::nio::CharArrayBuffer::length** [protected]
- 6.182.3.3 **int decaf::internal::nio::CharArrayBuffer::offset** [protected]
- 6.182.3.4 **bool decaf::internal::nio::CharArrayBuffer::readOnly** [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**CharArrayBuffer.h**

### 6.183 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:

```
#include <src/main/decaf/nio/CharBuffer.h>
```

Inheritance diagram for decaf::nio::CharBuffer:

#### Public Member Functions

- virtual **~CharBuffer** ()
- virtual std::string **toString** () const
- **CharBuffer & append** (char value) throw ( BufferOverflowException, ReadOnlyBufferException )  
*Appends the specified character to this buffer.*
- **CharBuffer & append** (const lang::CharSequence \*value) throw ( BufferOverflowException, ReadOnlyBufferException )  
*Appends the specified character sequence to this buffer.*
- **CharBuffer & append** (const lang::CharSequence \*value, int start, int end) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException )  
*Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.*
- virtual char \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the character array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

- virtual **CharBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only char buffer that shares this buffer's content.*
- char **charAt** (int index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads the character at the given index relative to the current position.*
- virtual **CharBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **CharBuffer** \* **duplicate** ()=0  
*Creates a new char buffer that shares this buffer's content.*
- virtual char **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual char **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **CharBuffer** & **get** (std::vector< char > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **CharBuffer** & **get** (char \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible char array.*
- int **length** () const  
*Returns the length of this character buffer.*
- **CharBuffer** & **put** (**CharBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the chars remaining in the given source buffer into this buffer.*
- **CharBuffer** & **put** (const char \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers chars into this buffer from the given source array.*
- **CharBuffer** & **put** (std::vector< char > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source char array into this buffer.*
- virtual **CharBuffer** & **put** (char value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given char into this buffer at the current position, and then increments the position.*
- virtual **CharBuffer** & **put** (int index, char value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given char into this buffer at the given index.*
- **CharBuffer** & **put** (std::string &src, int start, int end) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Relative bulk put method (optional operation).*

- **CharBuffer & put** (const std::string &src) throw ( BufferOverflowException, ReadOnlyBufferException )

*Relative bulk put method (optional operation).*

- virtual int **read** (**CharBuffer** \*target) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException )

*Attempts to read characters into the specified character buffer.*

- virtual **lang::CharSequence** \* **subSequence** (int start, int end) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.*

- virtual **CharBuffer** \* **slice** () const =0

*Creates a new **CharBuffer** (p. 1089) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **CharBuffer** &value) const

- virtual bool **equals** (const **CharBuffer** &value) const

- virtual bool **operator==** (const **CharBuffer** &value) const

- virtual bool **operator<** (const **CharBuffer** &value) const

## Static Public Member Functions

- static **CharBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Allocates a new character buffer.*

- static **CharBuffer** \* **wrap** (char \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Wraps the passed buffer with a new **CharBuffer** (p. 1089).*

- static **CharBuffer** \* **wrap** (std::vector< char > &buffer)

*Wraps the passed STL char Vector in a **CharBuffer** (p. 1089).*

## Protected Member Functions

- **CharBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **CharBuffer** (p. 1089) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.183.1 Detailed Description

This class defines four categories of operations upon character buffers:

- o Absolute and relative get and put methods that read and write single characters;
- o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and
- o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer.
- o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the CharSequence interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package `decaf.util.regex`.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

```
cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);
```

can, for example, be replaced by the single statement

```
cb.put("text/").put(subtype).put("; charset=").put(enc);
```

### 6.183.2 Constructor & Destructor Documentation

**6.183.2.1** `decaf::nio::CharBuffer::CharBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` `[protected]`

Creates a **CharBuffer** (p. 1089) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size of the array, this is the limit we read and write to.
-----------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if <i>capacity</i> is negative.
---------------------------------	---------------------------------

**6.183.2.2** `virtual decaf::nio::CharBuffer::~~CharBuffer ( )` `[inline, virtual]`

### 6.183.3 Member Function Documentation

6.183.3.1 **static CharBuffer\*** `decaf::nio::CharBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Allocates a new character buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

#### Parameters

<i>capacity</i>	The size of the Char buffer in chars ( 1 byte ).
-----------------	--

#### Returns

the **CharBuffer** (p. 1089) that was allocated, caller owns.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if capacity is negative.
----------------------------------	--------------------------

6.183.3.2 **CharBuffer&** `decaf::nio::CharBuffer::append ( const lang::CharSequence * value, int start, int end ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException )` [virtual]

Appends a subsequence of the specified character sequence to this buffer. If value is Null the the string "null" is appended to the buffer.

#### Parameters

<i>value</i>	The CharSequence to append.
<i>start</i>	The index to start appending from.
<i>end</i>	The index to append to.

#### Returns

a reference to this modified **CharBuffer** (p. 1089).

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is no more space
<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>IndexOutOfBoundsException</i>	if start > end, or > length of sequence.

Implements **decaf::lang::Appendable** (p. 695).

### 6.183.3.3 CharBuffer& decaf::nio::CharBuffer::append ( char *value* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** ) [virtual]

Appends the specified character to this buffer.

#### Parameters

<i>value</i>	The char to append.
--------------	---------------------

#### Returns

a reference to this modified **CharBuffer** (p. 1089).

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is no more space
<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.

Implements **decaf::lang::Appendable** (p. 694).

### 6.183.3.4 CharBuffer& decaf::nio::CharBuffer::append ( const lang::CharSequence \* *value* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** ) [virtual]

Appends the specified character sequence to this buffer.

If value is Null the the string "null" is appended to the buffer.

#### Parameters

<i>value</i>	The CharSequence to append.
--------------	-----------------------------

#### Returns

a reference to this modified **CharBuffer** (p. 1089)

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is no more space
<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.

Implements **decaf::lang::Appendable** (p. 695).

6.183.3.5 `virtual char* decaf::nio::CharBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 887).

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1083).

6.183.3.6 `virtual int decaf::nio::CharBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1083).



**6.183.3.7** `virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer ( ) const` `[pure virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only char buffer which the caller then owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1084).

**6.183.3.8** `char decaf::nio::CharBuffer::charAt ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` `[virtual]`

Reads the character at the given index relative to the current position.

### Parameters

<i>index</i>	- The index of the character to be read relative to position
--------------	--

### Returns

The character at index **position()** (p. 892) + index.

### Exceptions

<i>IndexOutOfBoundsException</i>	if the index + the current position exceeds the size of the buffer or the index is negative.
----------------------------------	--

Implements **decaf::lang::CharSequence** (p. 1108).

**6.183.3.9** `virtual CharBuffer& decaf::nio::CharBuffer::compact ( ) throw ( ReadOnlyBufferException )` `[pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index **limit()** (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **CharBuffer** (p. 1089).

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	- If this buffer is read-only
--	-------------------------------

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1084).

6.183.3.10 `virtual int decaf::nio::CharBuffer::compareTo ( const CharBuffer & value ) const`  
[virtual]

6.183.3.11 `virtual CharBuffer* decaf::nio::CharBuffer::duplicate ( )` [pure virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new char **Buffer** (p. 887) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1085).

6.183.3.12 `virtual bool decaf::nio::CharBuffer::equals ( const CharBuffer & value ) const`  
[virtual]

6.183.3.13 `virtual char decaf::nio::CharBuffer::get ( ) throw ( BufferUnderflowException )`  
[pure virtual]

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

### Returns

the char at the current position.

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return
--	---------------------------------

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1085).

6.183.3.14 `virtual char decaf::nio::CharBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the char at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the char is to be read.
--------------	---

#### Returns

the char that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit or is negative.
----------------------------------	---

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1085).

6.183.3.15 `CharBuffer& decaf::nio::CharBuffer::get ( std::vector< char > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

#### Returns

a reference to this **CharBuffer** (p. 1089).

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length chars remaining in this buffer.
--	--

6.183.3.16 **CharBuffer& decaf::nio::CharBuffer::get** ( *char \* buffer*, *int size*, *int offset*, *int length* ) throw ( **BufferUnderflowException**, **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException** )

Relative bulk get method.

This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 892), then no bytes are transferred and a **BufferUnderflowException** (p. 916) is thrown.

Otherwise, this method copies `length` chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 887).

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there are fewer than <code>length</code> chars remaining in this buffer
<b>NullPointerException</b>	if the passed buffer is null.
<b>IndexOutOfBoundsException</b>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

6.183.3.17 **virtual bool decaf::nio::CharBuffer::hasArray** ( ) const [pure virtual]

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1086).

6.183.3.18 `int decaf::nio::CharBuffer::length ( ) const [inline, virtual]`

Returns the length of this character buffer.

#### Returns

the length of this buffer from the position to the limit.

Implements **decaf::lang::CharSequence** (p. 1108).

6.183.3.19 `virtual bool decaf::nio::CharBuffer::operator< ( const CharBuffer & value ) const [virtual]`

6.183.3.20 `virtual bool decaf::nio::CharBuffer::operator== ( const CharBuffer & value ) const [virtual]`

6.183.3.21 `CharBuffer& decaf::nio::CharBuffer::put ( CharBuffer & src )  
throw ( BufferOverflowException, ReadOnlyBufferException,  
decaf::lang::exceptions::IllegalArgumentException )`

This method transfers the chars remaining in the given source buffer into this buffer.

If there are more chars remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 892), then no chars are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `n = src.remaining()` chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<code>src</code>	- the buffer to take chars from an place in this one.
------------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer for the remaining chars in the source buffer.
<b>IllegalArgumentException</b>	if the source buffer is this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

6.183.3.22 **CharBuffer& decaf::nio::CharBuffer::put** ( const char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException**, **decaf::lang::exceptions::IndexOutOfBoundsException**, **decaf::lang::exceptions::NullPointerException** )

This method transfers chars into this buffer from the given source array.

If there are more chars to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 892), then no chars are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The array from which chars are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of chars to be read from the given array.

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only
<b>NullPointerException</b>	if the passed buffer is null.
<b>IndexOutOfBoundsException</b>	if the preconditions of size, offset, or length are not met.

6.183.3.23 **virtual CharBuffer& decaf::nio::CharBuffer::put** ( char *value* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** ) [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The char value to be written.
--------------	-------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not smaller than its limit
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1086).

6.183.3.24 `virtual CharBuffer& decaf::nio::CharBuffer::put ( int index, char value )  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
ReadOnlyBufferException ) [pure virtual]`

Writes the given char into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The char to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1087).

6.183.3.25 `CharBuffer& decaf::nio::CharBuffer::put ( std::string & src, int start, int  
end ) throw ( BufferOverflowException, ReadOnlyBufferException,  
decaf::lang::exceptions::IndexOutOfBoundsException )`

Relative bulk put method (optional operation).

This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if `end - start > remaining()` (p. 892), then no characters are transferred and a **BufferOverflowException** (p. 914) is thrown.

**Returns**

a reference to this buffer

Otherwise, this method copies  $n = \text{end} - \text{start}$  characters from the given string into this buffer, starting at the given start index and at the current position of this buffer. The position of this buffer is then incremented by  $n$ .

**Parameters**

<i>src</i>	The string to copy from.
<i>start</i>	The position in <i>src</i> to start from.
<i>end</i>	The position in <i>src</i> to stop at.

**Returns**

a reference to this **CharBuffer** (p. 1089).

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not
<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only

#### 6.183.3.26 **CharBuffer& decaf::nio::CharBuffer::put ( const std::string & src ) throw ( BufferOverflowException, ReadOnlyBufferException )**

Relative bulk put method (optional operation).

This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form `dst.put(s)` behaves in exactly the same way as the invocation.

**Parameters**

<i>src</i>	The string to copy from.
------------	--------------------------

**Returns**

a reference to this **CharBuffer** (p. 1089).

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not.
---	---



<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.
--	------------------------------

#### 6.183.3.27 CharBuffer& decaf::nio::CharBuffer::put ( std::vector< char > & *buffer* ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source char array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>CharBuffer</b> (p. 1089).
---------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 914)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

#### 6.183.3.28 virtual int decaf::nio::CharBuffer::read ( CharBuffer \* *target* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException ) [virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

#### Parameters

<i>target</i>	The buffer to read characters into
---------------	------------------------------------

#### Returns

The number of characters added to the buffer, or string::npos if this source of characters is at its end

#### Exceptions

<i>NullPointerException</i>	if target is Null.
-----------------------------	--------------------

<i>IllegalArgumentException</i>	if target is this <b>CharBuffer</b> (p. 1089).
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is in read-only mode.

6.183.3.29 `virtual CharBuffer* decaf::nio::CharBuffer::slice ( ) const [pure virtual]`

Creates a new **CharBuffer** (p. 1089) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **CharBuffer** (p. 1089) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1088).

6.183.3.30 `virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence ( int start, int end ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 892) + start, and its limit will be **position()** (p. 892) + end. The new **Buffer** (p. 887) will be read-only if, and only if, this buffer is read-only.

#### Parameters

<i>start</i>	The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than <b>remaining()</b> (p. 892).
<i>end</i>	The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than <b>remaining()</b> (p. 892).

**Returns**

The new character buffer, caller owns.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the preconditions on start and end fail.
----------------------------------	---

Implements **decaf::lang::CharSequence** (p. 1109).

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1088).

6.183.3.31 `virtual std::string decaf::nio::CharBuffer::toString ( ) const` [virtual]

**Returns**

a std::string describing this object

Implements **decaf::lang::CharSequence** (p. 1109).

6.183.3.32 `static CharBuffer* decaf::nio::CharBuffer::wrap ( char * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **CharBuffer** (p. 1089).

The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the array passed in.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

**Returns**

a new **CharBuffer** (p. 1089) that is backed by buffer, caller owns.

**Exceptions**

<i>NullPointerException</i>	if the array pointer is Null.
<i>IndexOutOfBoundsException</i>	if capacity is negative.

6.183.3.33 **static CharBuffer\*** decaf::nio::CharBuffer::wrap ( std::vector< char > & *buffer* )  
[static]

Wraps the passed STL char Vector in a **CharBuffer** (p. 1089).

The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

#### Returns

a new **CharBuffer** (p. 1089) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**CharBuffer.h**

## 6.184 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 1107) is a readable sequence of char values.

```
#include <src/main/decaf/lang/CharSequence.h>
```

Inheritance diagram for decaf::lang::CharSequence:

#### Public Member Functions

- virtual ~**CharSequence** ()
- virtual int **length** () const =0
- virtual char **charAt** (int index) const =0 throw ( lang::exceptions::IndexOutOfBoundsException )

*Returns the Char at the specified index so long as the index is not greater than the length of the sequence.*

- virtual **CharSequence** \* **subSequence** (int start, int end) const =0 throw ( lang::exceptions::IndexOutOfBoundsException )

*Returns a new **CharSequence** (p. 1107) that is a subsequence of this sequence.*

- virtual std::string **toString** () const =0

### 6.184.1 Detailed Description

A **CharSequence** (p. 1107) is a readable sequence of char values.

This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 1107) should implement comparable, it is therefore up to the derived classes that implement this interface to define equality, which implies that comparison of two CharSequences does not have a contract on equality.

### 6.184.2 Constructor & Destructor Documentation

6.184.2.1 `virtual decaf::lang::CharSequence::~~CharSequence ( ) [inline, virtual]`

### 6.184.3 Member Function Documentation

6.184.3.1 `virtual char decaf::lang::CharSequence::charAt ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

#### Parameters

<i>index</i>	- position to return the char at.
--------------	-----------------------------------

#### Returns

the char at the given position

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > than <b>length()</b> (p. 1108) or negative
----------------------------------	--

Implemented in **decaf::lang::String** (p. 3611), and **decaf::nio::CharBuffer** (p. 1096).

6.184.3.2 `virtual int decaf::lang::CharSequence::length ( ) const [pure virtual]`

#### Returns

the length of the underlying character sequence.

Implemented in **decaf::lang::String** (p. 3612), and **decaf::nio::CharBuffer** (p. 1100).

6.184.3.3 `virtual CharSequence* decaf::lang::CharSequence::subSequence ( int start,  
int end ) const throw ( lang::exceptions::IndexOutOfBoundsException )`  
[pure virtual]

Returns a new **CharSequence** (p. 1107) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index `end - 1`. The length (in chars) of the returned sequence is `end - start`, so if `start == end` then an empty sequence is returned.

#### Parameters

<i>start</i>	- the start index, inclusive
<i>end</i>	- the end index, exclusive

#### Returns

a new **CharSequence** (p. 1107)

#### Exceptions

<i>IndexOutOfBoundsException</i>	if <code>start</code> or <code>end</code> > <b>length()</b> (p. 1108) or <code>start</code> or <code>end</code> are negative.
----------------------------------	---

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 1088), **decaf::lang::String** (p. 3612), and **decaf::nio::CharBuffer** (p. 1105).

6.184.3.4 `virtual std::string decaf::lang::CharSequence::toString ( ) const` [pure virtual]

#### Returns

the string representation of this **CharSequence** (p. 1107)

Implemented in **decaf::lang::String** (p. 3613), and **decaf::nio::CharBuffer** (p. 1106).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/CharSequence.h`

## 6.185 decaf::util::zip::CheckedInputStream Class Reference

An implementation of a `FilterInputStream` that will maintain a **Checksum** (p. 1114) of the bytes read, the **Checksum** (p. 1114) can then be used to verify the integrity of the input stream.

```
#include <src/main/decaf/util/zip/CheckedInputStream.h>
```

Inheritance diagram for `decaf::util::zip::CheckedInputStream`:

## Public Member Functions

- **Checksum** (InputStream \*inputStream, Checksum \*sum, bool own=false)

Create a new instance of a **Checksum** (p. 1109).

- virtual ~**Checksum** ()
- **Checksum** \* getChecksum () const

Returns a Pointer to the **Checksum** (p. 1114) that is in use by this **Checksum** (p. 1109).

- virtual long long **skip** (long long num) throw ( decaf::io::IOException )

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

### Parameters

num	The number of bytes to skip.
-----	------------------------------

### Returns

total bytes skipped

### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
UnsupportedOperationException	if the concrete stream class does not support skipping bytes.

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.185.1 Detailed Description

An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1114) of the bytes read, the **Checksum** (p. 1114) can then be used to verify the integrity of the input stream.

### Since

1.0

### 6.185.2 Constructor & Destructor Documentation

6.185.2.1 `decaf::util::zip::CheckedInputStream::CheckedInputStream ( InputStream * inputStream, Checksum * sum, bool own = false )`

Create a new instance of a **CheckedInputStream** (p. 1109).

#### Parameters

<i>inputStream</i>	The <b>InputStream</b> instance to Wrap.
<i>sum</i>	The <b>Checksum</b> (p. 1114) instance to use (does not take ownership of the Pointer).
<i>own</i>	Indicates if this filter should take ownership of the <b>InputStream</b> .

#### Exceptions

<i>NullPointerException</i>	if the <b>Checksum</b> (p. 1114) pointer is NULL.
-----------------------------	---

6.185.2.2 `virtual decaf::util::zip::CheckedInputStream::~~CheckedInputStream ( )`  
[virtual]

### 6.185.3 Member Function Documentation

6.185.3.1 `virtual int decaf::util::zip::CheckedInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.185.3.2 `virtual int decaf::util::zip::CheckedInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.185.3.3 `Checksum* decaf::util::zip::CheckedInputStream::getChecksum ( ) const`  
[inline]

Returns a Pointer to the **Checksum** (p. 1114) that is in use by this **CheckedInputStream** (p. 1109).

#### Returns

the pointer to the **Checksum** (p. 1114) instance that is in use by this object.



6.185.3.4 virtual long long decaf::util::zip::CheckedInputStream::skip ( long long *num* ) throw ( decaf::io::IOException ) [virtual]

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p.2002) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Adds the skipped bytes into the **Checksum** (p. 1114).

Reimplemented from **decaf::io::FilterInputStream** (p. 1860).

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**CheckedInputStream.h**

## 6.186 decaf::util::zip::CheckedOutputStream Class Reference

An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1114) of the bytes written, the **Checksum** (p. 1114) can then be used to verify the integrity of the output stream.

```
#include <src/main/decaf/util/zip/CheckedOutputStream.h>
```

Inheritance diagram for decaf::util::zip::CheckedOutputStream:

#### Public Member Functions

- **CheckedOutputStream** (decaf::io::OutputStream \*outputStream, Checksum

\*sum, bool **own**=false)

Create a new instance of a **CheckedOutputStream** (p. 1112).

- virtual **~CheckedOutputStream** ()
- **Checksum** \* **getChecksum** () const

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.186.1 Detailed Description

An implementation of a **FilterOutputStream** that will maintain a **Checksum** (p. 1114) of the bytes written, the **Checksum** (p. 1114) can then be used to verify the integrity of the output stream.

#### Since

1.0

### 6.186.2 Constructor & Destructor Documentation

6.186.2.1 decaf::util::zip::CheckedOutputStream::CheckedOutputStream ( decaf::io::OutputStream \* *outputStream*, **Checksum** \* *sum*, bool *own* = false )

Create a new instance of a **CheckedOutputStream** (p. 1112).

#### Parameters

<i>output-Stream</i>	The OutputStream instance to Wrap.
<i>sum</i>	The <b>Checksum</b> (p. 1114) instance to use (does not take ownership of the Pointer).
<i>own</i>	Indicates if this filer should take ownership of the InputStream.

#### Exceptions

<i>NullPointerException</i>	if the <b>Checksum</b> (p. 1114) pointer is NULL.
-----------------------------	---

6.186.2.2 virtual decaf::util::zip::CheckedOutputStream::~~CheckedOutputStream ( )  
[virtual]

### 6.186.3 Member Function Documentation

6.186.3.1 virtual void decaf::util::zip::ChecksumOutputStream::doWriteArrayBounded ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
[protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.186.3.2 virtual void decaf::util::zip::ChecksumOutputStream::doWriteByte ( unsigned char *value* ) throw ( decaf::io::IOException ) [protected, virtual]

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.186.3.3 **Checksum\*** decaf::util::zip::ChecksumOutputStream::getChecksum ( ) const  
[inline]

#### Returns

a pointer to the **Checksum** (p. 1114) instance in use by this object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**ChecksumOutputStream.h**

## 6.187 decaf::util::zip::Checksum Class Reference

An interface used to represent **Checksum** (p. 1114) values in the Zip package.

```
#include <src/main/decaf/util/zip/Checksum.h>
```

Inheritance diagram for decaf::util::zip::Checksum:

### Public Member Functions

- virtual **~Checksum** ()
- virtual long long **getValue** () const =0
- virtual void **reset** ()=0  
*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)=0  
*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

*Updates the current checksum with the specified array of bytes.*

- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length)=0  
throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Updates the current checksum with the specified array of bytes.*

- virtual void **update** (int byte)=0

*Updates the current checksum with the specified byte value.*

### 6.187.1 Detailed Description

An interface used to represent **Checksum** (p. 1114) values in the Zip package.

#### Since

1.0

### 6.187.2 Constructor & Destructor Documentation

6.187.2.1 virtual decaf::util::zip::Checksum::~Checksum ( ) [inline, virtual]

### 6.187.3 Member Function Documentation

6.187.3.1 virtual long long decaf::util::zip::Checksum::getValue ( ) const [pure virtual]

#### Returns

the current checksum value.

Implemented in **decaf::util::zip::Adler32** (p. 692), and **decaf::util::zip::CRC32** (p. 1491).

6.187.3.2 virtual void decaf::util::zip::Checksum::reset ( ) [pure virtual]

Reset the checksum to its initial value.

Implemented in **decaf::util::zip::Adler32** (p. 692), and **decaf::util::zip::CRC32** (p. 1491).

6.187.3.3 virtual void decaf::util::zip::Checksum::update ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]

Updates the current checksum with the specified array of bytes.

#### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

**Exceptions**

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.

Implemented in **decaf::util::zip::Adler32** (p. 692), and **decaf::util::zip::CRC32** (p. 1491).

**6.187.3.4** `virtual void decaf::util::zip::Checksum::update ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Updates the current checksum with the specified array of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.
----------------------------------	--

Implemented in **decaf::util::zip::Adler32** (p. 693), and **decaf::util::zip::CRC32** (p. 1491).

**6.187.3.5** `virtual void decaf::util::zip::Checksum::update ( const std::vector< unsigned char > & buffer ) [pure virtual]`

Updates the current checksum with the specified vector of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implemented in **decaf::util::zip::Adler32** (p. 693), and **decaf::util::zip::CRC32** (p. 1492).

**6.187.3.6** `virtual void decaf::util::zip::Checksum::update ( int byte ) [pure virtual]`

Updates the current checksum with the specified byte value.

**Parameters**

<i>byte</i>	The byte value to update the current <b>Checksum</b> (p. 1114) with (0..255).
-------------	---

Implemented in **decaf::util::zip::Adler32** (p. 693), and **decaf::util::zip::CRC32** (p. 1492).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/Checksum.h`

## 6.188 decaf::lang::exceptions::ClassCastException Class Reference

```
#include <src/main/decaf/lang/exceptions/ClassCastException.h>
```

Inheritance diagram for `decaf::lang::exceptions::ClassCastException`:

**Public Member Functions**

- **ClassCastException** () throw ()  
*Default Constructor.*
- **ClassCastException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **ClassCastException** (const **ClassCastException** &ex) throw ()  
*Copy Constructor.*
- **ClassCastException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **ClassCastException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ClassCastException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ClassCastException** \* **clone** () const  
*Clones this exception.*
- virtual ~**ClassCastException** () throw ()

### 6.188.1 Constructor & Destructor Documentation

6.188.1.1 `decaf::lang::exceptions::ClassCastException::ClassCastException ( ) throw ()`  
[inline]

Default Constructor.

6.188.1.2 `decaf::lang::exceptions::ClassCastException::ClassCastException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.188.1.3 `decaf::lang::exceptions::ClassCastException::ClassCastException ( const ClassCastException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.188.1.4 `decaf::lang::exceptions::ClassCastException::ClassCastException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.188.1.5 `decaf::lang::exceptions::ClassCastException::ClassCastException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.188.1.6 `decaf::lang::exceptions::ClassCastException::ClassCastException ( const char * file,  
const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.188.1.7 `virtual decaf::lang::exceptions::ClassCastException::~~ClassCastException ( ) throw  
() [inline, virtual]`

### 6.188.2 Member Function Documentation

6.188.2.1 `virtual ClassCastException* decaf::lang::exceptions::ClassCastException::clone ( )const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**ClassCastException.h**

## 6.189 cms::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/cms/Closeable.h>
```

Inheritance diagram for cms::Closeable:



## Public Member Functions

- virtual **~Closeable** ()
- virtual void **close** ()=0 throw ( CMSEException )

*Closes this object and deallocates the appropriate resources.*

### 6.189.1 Detailed Description

Interface for a class that implements the close method.

A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

#### Since

1.0

### 6.189.2 Constructor & Destructor Documentation

6.189.2.1 virtual cms::Closeable::~~Closeable ( ) [inline, virtual]

### 6.189.3 Member Function Documentation

6.189.3.1 virtual void cms::Closeable::close ( ) throw ( CMSEException ) [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an error occurs while the resource is being closed.
-----------------------------------	--

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1042), **activemq::cmsutil::CachedProducer** (p. 1046), **activemq::cmsutil::PooledSession** (p. 2907), **activemq::commands::ActiveMQTempDestination** (p. 549), **activemq::core::ActiveMQConnection** (p. 250), **activemq::core::ActiveMQConsumer** (p. 286), **activemq::core::ActiveMQProducer** (p. 443), **activemq::core::ActiveMQQueueBrowser** (p. 458), **activemq::core::ActiveMQSession** (p. 489), **cms::Connection** (p. 1234), and **cms::Session** (p. 3309).

The documentation for this class was generated from the following file:

- src/main/cms/**Closeable.h**

## 6.190 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/decaf/io/Closeable.h>
```

Inheritance diagram for decaf::io::Closeable:

### Public Member Functions

- virtual **~Closeable** ()
- virtual void **close** ()=0 throw ( io::IOException )  
*Closes this object and deallocates the appropriate resources.*

#### 6.190.1 Detailed Description

Interface for a class that implements the close method.

#### 6.190.2 Constructor & Destructor Documentation

6.190.2.1 virtual decaf::io::Closeable::~~Closeable ( ) [inline, virtual]

#### 6.190.3 Member Function Documentation

6.190.3.1 virtual void decaf::io::Closeable::close ( ) throw ( io::IOException ) [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
---------------------------------	-----------------------------------

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3234), **activemq::transport::failover::FailoverTransport** (p. 1838), **activemq::transport::inactivity::InactivityMonitor** (p. 1965), **activemq::transport::io::Transport** (p. 2107), **activemq::transport::mock::MockTransport** (p. 2726), **activemq::transport::tcp::TcpTransport** (p. 3697), **activemq::transport::TransportFilter** (p. 3829), **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2852), **decaf::internal::io::StandardErrorOutputStream** (p. 3523), **decaf::internal::io::StandardOutputStream** (p. 3526), **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2813), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2834), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3693), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 3695), **decaf::io::BlockingByteArrayInputStream** (p. 802), **decaf::io::BufferedInputStream**

## 6.191 activemq::transport::failover::CloseTransportsTask Class Reference 1125

(p. 897), **decaf::io::FilterInputStream** (p. 1857), **decaf::io::FilterOutputStream** (p. 1863), **decaf::io::InputStream** (p. 2004), **decaf::io::InputStreamReader** (p. 2014), **decaf::io::OutputStream** (p. 2858), **decaf::io::OutputStreamWriter** (p. 2866), **decaf::net::Socket** (p. 3452), **decaf::util::logging::ConsoleHandler** (p. 1368), **decaf::util::logging::StreamHandler** (p. 3593), **decaf::util::zip::DeflaterOutputStream** (p. 1685), and **decaf::util::zip::InflaterInputStream** (p. 1998).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Closeable.h`

## 6.191 activemq::transport::failover::CloseTransportsTask Class Reference

```
#include <src/main/activemq/transport/failover/CloseTransportsTask.h>
```

Inheritance diagram for `activemq::transport::failover::CloseTransportsTask`:

### Public Member Functions

- **CloseTransportsTask** ()
- virtual **~CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)  
*Add a new **Transport** (p. 3819) to close.*
- virtual bool **isPending** () const  
*This Task is pending if there are transports in the Queue that need to be closed.*
- virtual bool **iterate** ()  
*Return true until all transports have been closed and removed from the queue.*

### 6.191.1 Constructor & Destructor Documentation

6.191.1.1 `activemq::transport::failover::CloseTransportsTask::CloseTransportsTask ( )`

6.191.1.2 `virtual activemq::transport::failover::CloseTransportsTask::~~CloseTransportsTask ( ) [virtual]`

### 6.191.2 Member Function Documentation

6.191.2.1 `void activemq::transport::failover::CloseTransportsTask::add ( const Pointer< Transport > & transport )`

Add a new **Transport** (p. 3819) to close.

6.191.2.2 `virtual bool activemq::transport::failover::CloseTransportsTask::isPending ( ) const`  
`[virtual]`

This Task is pending if there are transports in the Queue that need to be closed.

#### Returns

true if there is a transport in the queue that needs closed.

Implements **activemq::threads::CompositeTask** (p. 1194).

6.191.2.3 `virtual bool activemq::transport::failover::CloseTransportsTask::iterate ( )`  
`[virtual]`

Return true until all transports have been closed and removed from the queue.

Implements **activemq::threads::Task** (p. 3679).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/CloseTransportsTask.h`

## 6.192 **activemq::cmsutil::CmsAccessor** Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p. 1140) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1294) to operate on.

```
#include <src/main/activemq/cmsutil/CmsAccessor.h>
```

Inheritance diagram for **activemq::cmsutil::CmsAccessor**:

#### Public Member Functions

- **CmsAccessor** ()
- virtual **~CmsAccessor** ()
- virtual **ResourceLifecycleManager** \* **getResourceLifecycleManager** ()
- virtual const **ResourceLifecycleManager** \* **getResourceLifecycleManager** () const
- virtual void **setConnectionFactory** (**cms::ConnectionFactory** \*connectionFactory)

*Set the ConnectionFactory to use for obtaining CMS Connections.*

- virtual const **cms::ConnectionFactory** \* **getConnectionFactory** () const

*Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*

- virtual **cms::ConnectionFactory** \* **getConnectionFactory** ()

*Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.*

- virtual void **setSessionAcknowledgeMode** (cms::Session::AcknowledgeMode sessionAcknowledgeMode)

*Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.*

- virtual cms::Session::AcknowledgeMode **getSessionAcknowledgeMode** () const

*Return the acknowledgment mode for CMS sessions.*

## Protected Member Functions

- **CmsAccessor** (const **CmsAccessor** &)
- **CmsAccessor** & **operator=** (const **CmsAccessor** &)
- virtual void **init** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )

*Initializes this object and prepares it for use.*

- virtual void **destroy** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )

*Shuts down this object and destroys any allocated resources.*

- virtual cms::Connection \* **createConnection** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )

*Create a CMS Connection via this template's ConnectionFactory.*

- virtual cms::Session \* **createSession** (cms::Connection \*con) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )

*Create a CMS Session for the given Connection.*

- virtual void **checkConnectionFactory** () throw ( decaf::lang::exceptions::IllegalStateException )

*Verifies that the connection factory is valid.*

### 6.192.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p. 1140) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1294) to operate on.

The subclass **activemq.cmsutil.CmsDestinationAccessor** (p. 1127) adds further, destination-related properties.

Not intended to be used directly.

#### See also

**activemq.cmsutil.CmsDestinationAccessor** (p. 1127)  
**activemq.cmsutil.CmsTemplate** (p. 1140)

### 6.192.2 Constructor & Destructor Documentation

6.192.2.1 `activemq::cmsutil::CmsAccessor::CmsAccessor ( const CmsAccessor & )`  
[inline, protected]

6.192.2.2 `activemq::cmsutil::CmsAccessor::CmsAccessor ( )`

6.192.2.3 `virtual activemq::cmsutil::CmsAccessor::~~CmsAccessor ( )` [virtual]

### 6.192.3 Member Function Documentation

6.192.3.1 `virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory ( ) throw ( decaf::lang::exceptions::IllegalStateException )` [protected, virtual]

Verifies that the connection factory is valid.

6.192.3.2 `virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection ( ) throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )` [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

#### Returns

the new CMS Connection

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1130)	if thrown by CMS API methods
---	------------------------------

6.192.3.3 `virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession ( cms::Connection * con ) throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )` [protected, virtual]

Create a CMS Session for the given Connection.

#### Parameters

<i>con</i>	the CMS Connection to create a Session for
------------	--

#### Returns

the new CMS Session

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	if thrown by CMS API methods
--	------------------------------

6.192.3.4 `virtual void activemq::cmsutil::CmsAccessor::destroy ( ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [inline, protected, virtual]`

Shuts down this object and destroys any allocated resources.

Reimplemented in **activemq::cmsutil::CmsDestinationAccessor** (p. 1129), and **activemq::cmsutil::CmsTemplate** (p. 1144).

6.192.3.5 `virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory ( ) const [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.192.3.6 `virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory ( ) [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.192.3.7 `virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager ( ) [inline, virtual]`

6.192.3.8 `virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager ( ) const [inline, virtual]`

6.192.3.9 `virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode ( ) const [inline, virtual]`

Return the acknowledgment mode for CMS sessions.

**Returns**

the acknowledgment mode applied by this accessor

6.192.3.10 `virtual void activemq::cmsutil::CmsAccessor::init ( ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )`  
`[inline, protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Reimplemented in **activemq::cmsutil::CmsDestinationAccessor** (p. 1129), and **activemq::cmsutil::CmsTemplate** (p. 1146).

6.192.3.11 `CmsAccessor& activemq::cmsutil::CmsAccessor::operator= ( const CmsAccessor & )` `[inline, protected]`

6.192.3.12 `virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory ( cms::ConnectionFactory * connectionFactory )` `[inline, virtual]`

Set the ConnectionFactory to use for obtaining CMS Connections.

6.192.3.13 `virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode ( cms::Session::AcknowledgeMode sessionAcknowledgeMode )`  
`[inline, virtual]`

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.

Default is AUTO\_ACKNOWLEDGE.

#### Parameters

<i>sessionAcknowledgeMode</i>	the acknowledgment mode
-------------------------------	-------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsAccessor.h`

## 6.193 activemq::cmsutil::CmsDestinationAccessor Class Reference

Extends the **CmsAccessor** (p. 1123) to add support for resolving destination names.

```
#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h>
```

Inheritance diagram for `activemq::cmsutil::CmsDestinationAccessor`:



## Public Member Functions

- **CmsDestinationAccessor** ()
- virtual **~CmsDestinationAccessor** ()
- virtual bool **isPubSubDomain** () const
- virtual void **setPubSubDomain** (bool pubSubDomain)
- virtual **DestinationResolver** \* **getDestinationResolver** ()
- virtual const **DestinationResolver** \* **getDestinationResolver** () const
- virtual void **setDestinationResolver** (**DestinationResolver** \*destRes)

## Protected Member Functions

- **CmsDestinationAccessor** (const **CmsDestinationAccessor** &)
- **CmsDestinationAccessor** & **operator=** (const **CmsDestinationAccessor** &)
- virtual void **init** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Initializes the destination resolver.*
- virtual void **destroy** () throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Calls **destroy()** (p. 1129) on the destination resolver.*
- virtual **cms::Destination** \* **resolveDestinationName** (**cms::Session** \*session, const std::string &destName) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
*Resolves the destination via the **DestinationResolver** (p. 1720).*
- virtual void **checkDestinationResolver** () throw ( decaf::lang::exceptions::IllegalStateException )  
*Verifies that the destination resolver is valid.*

### 6.193.1 Detailed Description

Extends the **CmsAccessor** (p. 1123) to add support for resolving destination names.

Not intended to be used directly.

#### See also

- CmsTemplate** (p. 1140)
- CmsAccessor** (p. 1123)

### 6.193.2 Constructor & Destructor Documentation

6.193.2.1 **activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor** ( const **CmsDestinationAccessor** & ) [*inline, protected*]

6.193.2.2 **activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor** ( )

6.193.2.3 `virtual activemq::cmsutil::CmsDestinationAccessor::~~CmsDestinationAccessor ( )`  
`[virtual]`

### 6.193.3 Member Function Documentation

6.193.3.1 `virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver`  
`( ) throw ( decaf::lang::exceptions::IllegalStateException )`  
`[protected, virtual]`

Verifies that the destination resolver is valid.

6.193.3.2 `virtual void activemq::cmsutil::CmsDestinationAccessor::destroy ( ) throw (`  
`cms::CMSException, decaf::lang::exceptions::IllegalStateException )`  
`[protected, virtual]`

Calls **destroy()** (p. 1129) on the destination resolver.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 1126).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1144).

6.193.3.3 `virtual const DestinationResolver* ac-`  
`tivemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ( ) const`  
`[inline, virtual]`

6.193.3.4 `virtual DestinationResolver* ac-`  
`tivemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ( )`  
`[inline, virtual]`

6.193.3.5 `virtual void activemq::cmsutil::CmsDestinationAccessor::init ( ) throw (`  
`cms::CMSException, decaf::lang::exceptions::IllegalStateException )`  
`[protected, virtual]`

Initializes the destination resolver.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 1126).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1146).

6.193.3.6 `virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain ( ) const`  
`[inline, virtual]`

6.193.3.7 `CmsDestinationAccessor& ac-`  
`tivemq::cmsutil::CmsDestinationAccessor::operator= ( const`  
`CmsDestinationAccessor & ) [inline, protected]`

6.193.3.8 virtual cms::Destination\* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName ( cms::Session \* *session*, const std::string & *destName* ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException )  
[protected, virtual]

Resolves the destination via the **DestinationResolver** (p. 1720).

#### Parameters

<i>session</i>	the session
<i>destName</i>	the name of the destination.

#### Returns

the destination

#### Exceptions

<b>cms::CMSException</b> (p. 1130)	if resolution failed.
<b>decaf::lang::exceptions</b> (p. 1959)	if the destination resolver property is NULL.

6.193.3.9 virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver ( DestinationResolver \* *destRes* ) [inline, virtual]

6.193.3.10 virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain ( bool *pubSubDomain* ) [inline, virtual]

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 1152).

Referenced by **activemq::cmsutil::CmsTemplate::setPubSubDomain()**.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsDestinationAccessor.h**

## 6.194 cms::CMSException Class Reference

CMS API Exception that is the base for all exceptions thrown from CMS classes.

```
#include <src/main/cms/CMSException.h>
```

Inheritance diagram for cms::CMSException:

## Public Member Functions

- **CMSEException** () throw ()
- **CMSEException** (const **CMSEException** &ex) throw ()
- **CMSEException** (const std::string &message, const std::exception \*cause) throw ()
- **CMSEException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSEException** () throw ()
- virtual std::string **getMessage** () const  
*Gets the cause of the error.*
- virtual const std::exception \* **getCause** () const  
*Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **setMark** (const char \*file, const int lineNumber)  
*Adds a file/line number to the stack trace.*
- virtual void **printStackTrace** () const  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const  
*Gets the stack trace as one contiguous string.*
- virtual const char \* **what** () const throw ()  
*Overloads the std::exception **what()** (p. 1133) function to return the cause of the exception.*

### 6.194.1 Detailed Description

CMS API Exception that is the base for all exceptions thrown from CMS classes.

This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an std::exception.

Since the contained cause exception is of type std::exception and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSEException** (p. 1130). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSEException** (p. 1130) instances.

Since

1.0

## 6.194.2 Constructor & Destructor Documentation

6.194.2.1 cms::CMSEException::CMSEException ( ) throw ()

6.194.2.2 cms::CMSEException::CMSEException ( const CMSEException & ex ) throw ()

6.194.2.3 cms::CMSEException::CMSEException ( const std::string & message, const std::exception \* cause ) throw ()

6.194.2.4 cms::CMSEException::CMSEException ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.194.2.5 virtual cms::CMSEException::~~CMSEException ( ) throw () [virtual]

## 6.194.3 Member Function Documentation

6.194.3.1 virtual const std::exception\* cms::CMSEException::getCause ( ) const [virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

### Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

6.194.3.2 virtual std::string cms::CMSEException::getMessage ( ) const [virtual]

Gets the cause of the error.

### Returns

string errors message

6.194.3.3 virtual std::vector< std::pair< std::string, int> > cms::CMSEException::getStackTrace ( ) const [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

### Returns

vector containing stack trace strings

6.194.3.4 `virtual std::string cms::CMSException::getStackTraceString ( ) const`  
`[virtual]`

Gets the stack trace as one contiguous string.

#### Returns

string with formatted stack trace data

6.194.3.5 `virtual void cms::CMSException::printStackTrace ( ) const` `[virtual]`

Prints the stack trace to `std::err`.

6.194.3.6 `virtual void cms::CMSException::printStackTrace ( std::ostream & stream ) const`  
`[virtual]`

Prints the stack trace to the given output stream.

#### Parameters

<i>stream</i>	the target output stream.
---------------	---------------------------

6.194.3.7 `virtual void cms::CMSException::setMark ( const char * file, const int lineNumber )`  
`[virtual]`

Adds a file/line number to the stack trace.

#### Parameters

<i>file</i>	The name of the file calling this method (use <code>__FILE__</code> ).
<i>lineNumber</i>	The line number in the calling file (use <code>__LINE__</code> ).

6.194.3.8 `virtual const char* cms::CMSException::what ( ) const throw ()` `[virtual]`

Overloads the `std::exception` **what()** (p. 1133) function to return the cause of the exception.

#### Returns

const char pointer to error message

The documentation for this class was generated from the following file:

- `src/main/cms/CMSException.h`

## 6.195 activemq::util::CMSExceptionSupport Class Reference

```
#include <src/main/activemq/util/CMSExceptionSupport.h>
```

### Public Member Functions

- virtual `~CMSExceptionSupport()`

### Static Public Member Functions

- static `cms::CMSException create` (const std::string &msg, const `decaf::lang::Exception` &cause)
- static `cms::CMSException create` (const `decaf::lang::Exception` &cause)
- static `cms::MessageEOFException createMessageEOFException` (const `decaf::lang::Exception` &cause)
- static `cms::MessageFormatException createMessageFormatException` (const `decaf::lang::Exception` &cause)

### 6.195.1 Constructor & Destructor Documentation

6.195.1.1 virtual `activemq::util::CMSExceptionSupport::~~CMSExceptionSupport()` [virtual]

### 6.195.2 Member Function Documentation

6.195.2.1 static `cms::CMSException activemq::util::CMSExceptionSupport::create` ( const std::string & msg, const `decaf::lang::Exception` & cause ) [static]

6.195.2.2 static `cms::CMSException activemq::util::CMSExceptionSupport::create` ( const `decaf::lang::Exception` & cause ) [static]

6.195.2.3 static `cms::MessageEOFException activemq::util::CMSExceptionSupport::createMessageEOFException` ( const `decaf::lang::Exception` & cause ) [static]

6.195.2.4 static `cms::MessageFormatException activemq::util::CMSExceptionSupport::createMessageFormatException` ( const `decaf::lang::Exception` & cause ) [static]

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`

>::getShortProperty(), and activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty().

The documentation for this class was generated from the following file:

- src/main/activemq/util/**CMSExceptionSupport.h**

## 6.196 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

```
#include <src/main/cms/CMSProperties.h>
```

Inheritance diagram for cms::CMSProperties:

### Public Member Functions

- virtual **~CMSProperties** ()
- virtual bool **isEmpty** () const =0  
*Returns true if the properties object is empty.*
- virtual const char \* **getProperty** (const std::string &name) const =0  
*Looks up the value for the given property.*
- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const =0  
*Looks up the value for the given property.*
- virtual void **setProperty** (const std::string &name, const std::string &value)=0  
*Sets the value for a given property.*
- virtual bool **hasProperty** (const std::string &name) const =0  
*Check to see if the Property exists in the set.*
- virtual void **remove** (const std::string &name)=0  
*Removes the property with the given name.*
- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const =0  
*Method that serializes the contents of the property map to an array.*
- virtual void **copy** (const **CMSProperties** \*source)=0  
*Copies the contents of the given properties object to this one.*
- virtual **CMSProperties** \* **clone** () const =0  
*Clones this object.*
- virtual void **clear** ()=0  
*Clears all properties from the map.*
- virtual std::string **toString** () const =0  
*Formats the contents of the Properties Object into a string that can be logged, etc.*



### 6.196.1 Detailed Description

Interface for a Java-like properties object.

This is essentially a map of key-value string pairs.

#### Since

1.1

### 6.196.2 Constructor & Destructor Documentation

6.196.2.1 `virtual cms::CMSProperties::~~CMSProperties ( ) [inline, virtual]`

### 6.196.3 Member Function Documentation

6.196.3.1 `virtual void cms::CMSProperties::clear ( ) [pure virtual]`

Clears all properties from the map.

Implemented in **activemq::util::ActiveMQProperties** (p. 450).

6.196.3.2 `virtual CMSProperties* cms::CMSProperties::clone ( ) const [pure virtual]`

Clones this object.

#### Returns

a replica of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 450).

6.196.3.3 `virtual void cms::CMSProperties::copy ( const CMSProperties * source ) [pure virtual]`

Copies the contents of the given properties object to this one.

#### Parameters

<i>source</i>	The source properties object.
---------------	-------------------------------

6.196.3.4 `virtual const char* cms::CMSProperties::getProperty ( const std::string & name ) const [pure virtual]`

Looks up the value for the given property.

**Parameters**

<i>name</i>	The name of the property to be looked up.
-------------	---

**Returns**

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in **activemq::util::ActiveMQProperties** (p. 451).

6.196.3.5 `virtual std::string cms::CMSProperties::getProperty ( const std::string & name, const std::string & defaultValue ) const [pure virtual]`

Looks up the value for the given property.

**Parameters**

<i>name</i>	the name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

**Returns**

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implemented in **activemq::util::ActiveMQProperties** (p. 451).

6.196.3.6 `virtual bool cms::CMSProperties::hasProperty ( const std::string & name ) const [pure virtual]`

Check to see if the Property exists in the set.

**Parameters**

<i>name</i>	the name of the property to check
-------------	-----------------------------------

**Returns**

true if property exists, false otherwise.

Implemented in **activemq::util::ActiveMQProperties** (p. 451).

6.196.3.7 `virtual bool cms::CMSProperties::isEmpty ( ) const [pure virtual]`

Returns true if the properties object is empty.

**Returns**

true if empty

Implemented in **activemq::util::ActiveMQProperties** (p. 452).

6.196.3.8 `virtual void cms::CMSProperties::remove ( const std::string & name ) [pure virtual]`

Removes the property with the given name.

#### Parameters

<i>name</i>	the name of the property to be removed.s
-------------	--

Implemented in **activemq::util::ActiveMQProperties** (p. 452).

6.196.3.9 `virtual void cms::CMSProperties::setProperty ( const std::string & name, const std::string & value ) [pure virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

#### Parameters

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

Implemented in **activemq::util::ActiveMQProperties** (p. 452).

6.196.3.10 `virtual std::vector< std::pair< std::string, std::string > > cms::CMSProperties::toArray ( ) const [pure virtual]`

Method that serializes the contents of the property map to an array.

#### Returns

list of pairs where the first is the name and the second is the value.

Implemented in **activemq::util::ActiveMQProperties** (p. 452).

6.196.3.11 `virtual std::string cms::CMSProperties::toString ( ) const [pure virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

#### Returns

string value of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 453).

The documentation for this class was generated from the following file:

- src/main/cms/**CMSProperties.h**

## 6.197 cms::CMSSecurityException Class Reference

This exception must be thrown when a provider rejects a user name/password submitted by a client.

```
#include <src/main/cms/CMSSecurityException.h>
```

Inheritance diagram for cms::CMSSecurityException:

### Public Member Functions

- **CMSSecurityException** () throw ()
- **CMSSecurityException** (const **CMSSecurityException** &ex) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception \*cause) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSSecurityException** () throw ()

### 6.197.1 Detailed Description

This exception must be thrown when a provider rejects a user name/password submitted by a client.

It may also be thrown for any case where a security restriction prevents a method from completing.

#### Since

1.3

### 6.197.2 Constructor & Destructor Documentation

6.197.2.1 cms::CMSSecurityException::CMSSecurityException ( ) throw ()

6.197.2.2 cms::CMSSecurityException::CMSSecurityException ( const CMSSecurityException & ex ) throw ()

6.197.2.3 cms::CMSSecurityException::CMSSecurityException ( const std::string & message, const std::exception \* cause ) throw ()

6.197.2.4 cms::CMSSecurityException::CMSSecurityException ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.197.2.5 virtual cms::CMSSecurityException::~~CMSSecurityException ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/CMSSecurityException.h

## 6.198 activemq::cmsutil::CmsTemplate Class Reference

**CmsTemplate** (p. 1140) simplifies performing synchronous CMS operations.

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate:

### Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

### Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (cms::ConnectionFactory \*connectionFactory)
- virtual ~**CmsTemplate** ()
- virtual void **setDefaultDestination** (cms::Destination \*defaultDestination)  
*Sets the destination object to be used by default for send/receive operations.*
- virtual const cms::Destination \* **getDefaultDestination** () const  
*Retrieves the default destination to be used for send/receive operations.*
- virtual cms::Destination \* **getDefaultDestination** ()  
*Retrieves the default destination to be used for send/receive operations.*
- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)  
*Sets the name of the default destination to be used from send/receive operations.*
- virtual const std::string **getDefaultDestinationName** () const  
*Gets the name of the default destination to be used for send/receive operations.*
- virtual void **setPubSubDomain** (bool pubSubDomain)  
*Indicates whether the default destination is a topic (true) or a queue (false).*
- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const

- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)
 

*Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.*
- virtual bool **isExplicitQosEnabled** () const
 

*If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.*
- virtual void **setDeliveryPersistent** (bool deliveryPersistent)
 

*Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").*
- virtual void **setDeliveryMode** (int deliveryMode)
 

*Set the delivery mode to use when sending a message.*
- virtual int **getDeliveryMode** () const
 

*Return the delivery mode to use when sending a message.*
- virtual void **setPriority** (int priority)
 

*Set the priority of a message when sending.*
- virtual int **getPriority** () const
 

*Return the priority of a message when sending.*
- virtual void **setTimeToLive** (long long timeToLive)
 

*Set the time-to-live of the message when sending.*
- virtual long long **getTimeToLive** () const
 

*Return the time-to-live of the message when sending.*
- virtual void **execute** (**SessionCallback** \*action) throw ( cms::CMSException )
 

*Executes the given action within a CMS Session.*
- virtual void **execute** (**ProducerCallback** \*action) throw ( cms::CMSException )
 

*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **execute** (cms::Destination \*dest, **ProducerCallback** \*action) throw ( cms::CMSException )
 

*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **execute** (const std::string &destinationName, **ProducerCallback** \*action) throw ( cms::CMSException )
 

*Executes the given action and provides it with a CMS Session and producer.*
- virtual void **send** (**MessageCreator** \*messageCreator) throw ( cms::CMSException )
 

*Convenience method for sending a message to the default destination.*
- virtual void **send** (cms::Destination \*dest, **MessageCreator** \*messageCreator) throw ( cms::CMSException )
 

*Convenience method for sending a message to the specified destination.*
- virtual void **send** (const std::string &destinationName, **MessageCreator** \*messageCreator) throw ( cms::CMSException )

*Convenience method for sending a message to the specified destination.*

- virtual **cms::Message \* receive** () throw ( cms::CMSEException )  
*Performs a synchronous read from the default destination.*
- virtual **cms::Message \* receive** (cms::Destination \*destination) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination.*
- virtual **cms::Message \* receive** (const std::string &destinationName) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination.*
- virtual **cms::Message \* receiveSelected** (const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read consuming only messages identified by the given selector.*
- virtual **cms::Message \* receiveSelected** (cms::Destination \*destination, const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.*
- virtual **cms::Message \* receiveSelected** (const std::string &destinationName, const std::string &selector) throw ( cms::CMSEException )  
*Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.*

### Static Public Attributes

- static const long long **RECEIVE\_TIMEOUT\_NO\_WAIT**  
*Timeout value indicating that a receive operation should check if a message is immediately available without blocking.*
- static const long long **RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT**  
*Timeout value indicating a blocking receive without timeout.*
- static const int **DEFAULT\_PRIORITY**  
*Default message priority.*
- static const long long **DEFAULT\_TIME\_TO\_LIVE**  
*My default, messages should live forever.*

### Protected Member Functions

- **CmsTemplate** (const **CmsTemplate** &)
- **CmsTemplate & operator=** (const **CmsTemplate** &)
- void **init** () throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )  
*Initializes this object and prepares it for use.*
- void **destroy** () throw ( cms::CMSEException, decaf::lang::exceptions::IllegalStateException )  
*Clears all internal resources.*

## Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

### 6.198.1 Detailed Description

**CmsTemplate** (p. 1140) simplifies performing synchronous CMS operations.

This class is intended to be for CMS what Spring's `JmsTemplate` is for JMS. Provided with a CMS `ConnectionFactory`, creates and manages all other CMS resources internally.

Before using **CmsTemplate** (p. 1140) the user must first set the destination (either by name or by setting the destination object directly) and then call `init` to initialize the object for use.

**CmsTemplate** (p. 1140) allows the user to get access to a CMS `Session` through a user-defined **SessionCallback** (p. 3319). Similarly, if the user wants direct access to a CMS `MessageProducer`, it can provide a **ProducerCallback** (p. 3012). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the `send` methods.

#### See also

- SessionCallback** (p. 3319)
- ProducerCallback** (p. 3012)
- MessageCreator** (p. 2554)

### 6.198.2 Constructor & Destructor Documentation

6.198.2.1 `activemq::cmsutil::CmsTemplate::CmsTemplate ( const CmsTemplate & )`  
[inline, protected]

6.198.2.2 `activemq::cmsutil::CmsTemplate::CmsTemplate ( )`

6.198.2.3 `activemq::cmsutil::CmsTemplate::CmsTemplate ( cms::ConnectionFactory * connectionFactory )`

6.198.2.4 `virtual activemq::cmsutil::CmsTemplate::~~CmsTemplate ( )` [virtual]

### 6.198.3 Member Function Documentation



6.198.3.1 `void activemq::cmsutil::CmsTemplate::destroy ( ) throw ( cms::CMSException, decaf::lang::exceptions::IllegalStateException ) [protected, virtual]`

Clears all internal resources.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1129).

6.198.3.2 `virtual void activemq::cmsutil::CmsTemplate::execute ( SessionCallback * action ) throw ( cms::CMSException ) [virtual]`

Executes the given action within a CMS Session.

#### Parameters

<i>action</i>	the action to perform within a CMS Session
---------------	--

#### Exceptions

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs.
--	----------------------------

6.198.3.3 `virtual void activemq::cmsutil::CmsTemplate::execute ( ProducerCallback * action ) throw ( cms::CMSException ) [virtual]`

Executes the given action and provides it with a CMS Session and producer.

#### Parameters

<i>action</i>	the action to perform
---------------	-----------------------

#### Exceptions

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs.
--	----------------------------

6.198.3.4 `virtual void activemq::cmsutil::CmsTemplate::execute ( cms::Destination * dest, ProducerCallback * action ) throw ( cms::CMSException ) [virtual]`

Executes the given action and provides it with a CMS Session and producer.

#### Parameters

<i>dest</i>	the destination to send messages to
<i>action</i>	the action to perform

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs.
--	----------------------------

6.198.3.5 `virtual void activemq::cmsutil::CmsTemplate::execute ( const std::string & destinationName, ProducerCallback * action ) throw ( cms::CMSException )`  
[virtual]

Executes the given action and provides it with a CMS Session and producer.

**Parameters**

<i>destination-Name</i>	the name of the destination to send messages to (to internally be resolved to an actual destination)
<i>action</i>	the action to perform

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs.
--	----------------------------

6.198.3.6 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination ( )` [inline, virtual]

Retrieves the default destination to be used for send/receive operations.

**Returns**

the default destination. Non-const version of this method.

6.198.3.7 `virtual const cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination ( ) const`  
[inline, virtual]

Retrieves the default destination to be used for send/receive operations.

**Returns**

the default destination. Const version of this method.

6.198.3.8 `virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName ( ) const` [inline, virtual]

Gets the name of the default destination to be used for send/receive operations.

The destination type (topic/queue) is determined by the `pubSubDomain` property.

### Returns

the default name of the destination for send/receive operations.

**6.198.3.9** `virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode ( ) const`  
[inline, virtual]

Return the delivery mode to use when sending a message.

**6.198.3.10** `virtual int activemq::cmsutil::CmsTemplate::getPriority ( ) const` [inline, virtual]

Return the priority of a message when sending.

**6.198.3.11** `virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout ( ) const`  
[inline, virtual]

**6.198.3.12** `virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive ( ) const`  
[inline, virtual]

Return the time-to-live of the message when sending.

**6.198.3.13** `void activemq::cmsutil::CmsTemplate::init ( ) throw ( cms::CMSException,  
decaf::lang::exceptions::IllegalStateException )` [protected, virtual]

Initializes this object and prepares it for use.

This should be called before any other methods are called.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1129).

**6.198.3.14** `virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled ( ) const`  
[inline, virtual]

If "true", then the values of `deliveryMode`, `priority`, and `timeToLive` will be used when sending a message.

Otherwise, the default values, that may be set administratively, will be used.

### Returns

true if overriding default values of QOS parameters (`deliveryMode`, `priority`, and `timeToLive`)

**See also****setDeliveryMode** (p. 1151)**setPriority** (p. 1152)**setTimeToLive** (p. 1153)

6.198.3.15 `virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled ( ) const`  
`[inline, virtual]`

6.198.3.16 `virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled ( )`  
`const [inline, virtual]`

6.198.3.17 `virtual bool activemq::cmsutil::CmsTemplate::isNoLocal ( ) const` `[inline,`  
`virtual]`

6.198.3.18 `CmsTemplate& activemq::cmsutil::CmsTemplate::operator= ( const`  
`CmsTemplate & ) [inline, protected]`

6.198.3.19 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ( ) throw (`  
`cms::CMSEException ) [virtual]`

Performs a synchronous read from the default destination.

**Returns**

the message

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1130)	thrown if an error occurs
---	---------------------------

6.198.3.20 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ( const`  
`std::string & destinationName ) throw ( cms::CMSEException ) [virtual]`

Performs a synchronous read from the specified destination.

**Parameters**

<i>destination-Name</i>	the name of the destination to receive on (will be resolved to destination internally).
-------------------------	---

**Returns**

the message

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs
--	---------------------------

6.198.3.21 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ( cms::Destination * destination ) throw ( cms::CMSException )`  
[virtual]

Performs a synchronous read from the specified destination.

**Parameters**

<i>destination</i>	the destination to receive on
--------------------	-------------------------------

**Returns**

the message

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs
--	---------------------------

6.198.3.22 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected ( const std::string & destinationName, const std::string & selector ) throw ( cms::CMSException )` [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

**Parameters**

<i>destination-Name</i>	the name of the destination to receive on (will be resolved to destination internally).
<i>selector</i>	the selector expression.

**Returns**

the message

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs
--	---------------------------

6.198.3.23 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected ( cms::Destination * destination, const std::string & selector ) throw ( cms::CMSEException )` [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

#### Parameters

<i>destination</i>	the destination to receive on.
<i>selector</i>	the selector expression.

#### Returns

the message

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1130)	thrown if an error occurs
---	---------------------------

6.198.3.24 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected ( const std::string & selector ) throw ( cms::CMSEException )` [virtual]

Performs a synchronous read consuming only messages identified by the given selector.

#### Parameters

<i>selector</i>	the selector expression.
-----------------	--------------------------

#### Returns

the message

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1130)	thrown if an error occurs
---	---------------------------

6.198.3.25 `virtual void activemq::cmsutil::CmsTemplate::send ( const std::string & destinationName, MessageCreator * messageCreator ) throw ( cms::CMSEException )` [virtual]

Convenience method for sending a message to the specified destination.

#### Parameters

<i>destination-Name</i>	The name of the destination to send to.
<i>message-Creator</i>	Responsible for creating the message to be sent

### Exceptions

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs.
--	----------------------------

6.198.3.26 virtual void activemq::cmsutil::CmsTemplate::send ( **MessageCreator** \* *messageCreator* ) throw ( **cms::CMSException** ) [virtual]

Convenience method for sending a message to the default destination.

### Parameters

<i>message-Creator</i>	Responsible for creating the message to be sent
------------------------	---

### Exceptions

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs.
--	----------------------------

6.198.3.27 virtual void activemq::cmsutil::CmsTemplate::send ( **cms::Destination** \* *dest*, **MessageCreator** \* *messageCreator* ) throw ( **cms::CMSException** ) [virtual]

Convenience method for sending a message to the specified destination.

### Parameters

<i>dest</i>	The destination to send to
<i>message-Creator</i>	Responsible for creating the message to be sent

### Exceptions

<b><i>cms::CMSException</i></b> (p. 1130)	thrown if an error occurs.
--	----------------------------

6.198.3.28 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination ( cms::Destination * defaultDestination ) [inline, virtual]`

Sets the destination object to be used by default for send/receive operations.

If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

#### Parameters

<i>defaultDestination</i>	the default destination
---------------------------	-------------------------

6.198.3.29 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName ( const std::string & defaultDestinationName ) [inline, virtual]`

Sets the name of the default destination to be used from send/receive operations.

Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

#### Parameters

<i>defaultDestinationName</i>	the name of the destination for send/receive to by default.
-------------------------------	---

6.198.3.30 `virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode ( int deliveryMode ) [inline, virtual]`

Set the delivery mode to use when sending a message.

Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

#### Parameters

<i>deliveryMode</i>	the delivery mode to use
---------------------	--------------------------

#### See also

**isExplicitQosEnabled** (p. 1146)

6.198.3.31 `virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent ( bool deliveryPersistent ) [inline, virtual]`

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").



This will set the delivery mode accordingly, to either "PERSISTENT" or "NON\_PERSISTENT".

Default it "true" aka delivery mode "PERSISTENT".

#### See also

**setDeliveryMode(int)** (p. 1151)

6.198.3.32 `virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled ( bool explicitQosEnabled ) [inline, virtual]`

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

#### See also

**setDeliveryMode** (p. 1151)

**setPriority** (p. 1152)

**setTimeToLive** (p. 1153)

6.198.3.33 `virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled ( bool messageIdEnabled ) [inline, virtual]`

6.198.3.34 `virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled ( bool messageTimestampEnabled ) [inline, virtual]`

6.198.3.35 `virtual void activemq::cmsutil::CmsTemplate::setNoLocal ( bool noLocal ) [inline, virtual]`

6.198.3.36 `virtual void activemq::cmsutil::CmsTemplate::setPriority ( int priority ) [inline, virtual]`

Set the priority of a message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

#### See also

**isExplicitQosEnabled** (p. 1146)

6.198.3.37 `virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain ( bool pubSubDomain ) [inline, virtual]`

Indicates whether the default destination is a topic (true) or a queue (false).

Calling this method will set the `defaultDestination` property to NULL.

#### Parameters

<i>pubSubDomain</i>	indicates whether to use pub-sub messaging (topics).
---------------------	--

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 1130).

References **activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain()**.

6.198.3.38 **virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout ( long long *receiveTimeout* )** [inline, virtual]

6.198.3.39 **virtual void activemq::cmsutil::CmsTemplate::setTimeToLive ( long long *timeToLive* )** [inline, virtual]

Set the time-to-live of the message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

#### Parameters

<i>timeToLive</i>	the message's lifetime (in milliseconds)
-------------------	--

#### See also

**isExplicitQosEnabled** (p. 1146)

### 6.198.4 Friends And Related Function Documentation

6.198.4.1 **friend class ProducerExecutor** [friend]

6.198.4.2 **friend class ReceiveExecutor** [friend]

6.198.4.3 **friend class ResolveProducerExecutor** [friend]

6.198.4.4 **friend class ResolveReceiveExecutor** [friend]

6.198.4.5 **friend class SendExecutor** [friend]

### 6.198.5 Field Documentation

6.198.5.1 **const int activemq::cmsutil::CmsTemplate::DEFAULT\_PRIORITY** [static]

Default message priority.

6.198.5.2 `const long long activemq::cmsutil::CmsTemplate::DEFAULT_TIME_TO_LIVE [static]`

My default, messages should live forever.

6.198.5.3 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_INDEFINITE_WAIT [static]`

Timeout value indicating a blocking receive without timeout.

6.198.5.4 `const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_NO_WAIT [static]`

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.199 code Struct Reference

```
#include <src/main/decaf/internal/util/zip/inftrees.h>
```

### Data Fields

- unsigned char **op**
- unsigned char **bits**
- unsigned short **val**

### 6.199.1 Field Documentation

6.199.1.1 unsigned char **code::bits**

6.199.1.2 unsigned char **code::op**

6.199.1.3 unsigned short **code::val**

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/inftrees.h`

## 6.200 decaf::util::Collection< E > Class Template Reference

The root interface in the collection hierarchy.

```
#include <src/main/decaf/util/Collection.h>
```

Inheritance diagram for decaf::util::Collection< E >:

### Public Member Functions

- virtual **~Collection** ()
- virtual bool **add** (const E &value)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Returns true if this collection changed as a result of the call.*
- virtual bool **addAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )  
*Adds all of the elements in the specified collection to this collection.*
- virtual void **clear** ()=0 throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all of the elements from this collection (optional operation).*
- virtual bool **contains** (const E &value) const =0 throw ( lang::Exception )  
*Returns true if this collection contains the specified element.*
- virtual bool **containsAll** (const **Collection**< E > &collection) const =0 throw ( lang::Exception )  
*Returns true if this collection contains all of the elements in the specified collection.*
- virtual bool **equals** (const **Collection**< E > &value) const =0  
*Compares the passed collection to this one, if they contain the same elements, i.e.*
- virtual bool **isEmpty** () const =0
- virtual bool **remove** (const E &value)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes a single instance of the specified element from the collection.*
- virtual bool **removeAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Removes all this collection's elements that are also contained in the specified collection (optional operation).*
- virtual bool **retainAll** (const **Collection**< E > &collection)=0 throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )  
*Retains only the elements in this collection that are contained in the specified collection (optional operation).*
- virtual std::size\_t **size** () const =0  
*Returns the number of elements in this collection.*
- virtual std::vector< E > **toArray** () const =0  
*Returns an array containing all of the elements in this collection.*

### 6.200.1 Detailed Description

```
template<typename E> class decaf::util::Collection< E >
```

The root interface in the collection hierarchy.

A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 1155) implementation classes (which typically implement **Collection** (p. 1155) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 1155), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 1155) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw `UnsupportedOperationException` if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an `UnsupportedOperationException` if the invocation would have no effect on the collection. For example, invoking the `addAll(Collection)` method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the `equals` method. For example, the specification for the `contains(Object o)` method says: "returns true if and only if this collection contains at least one element `e` such that (`o==null ? e==null : o.equals(e)`)."

Since

1.0

### 6.200.2 Constructor & Destructor Documentation

```
6.200.2.1 template<typename E> virtual decaf::util::Collection< E >::~~Collection ( )
[inline, virtual]
```

### 6.200.3 Member Function Documentation

```
6.200.3.1 template<typename E> virtual bool decaf::util::Collection< E >::add ( const E
& value ) throw ( lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException,
lang::exceptions::IllegalStateException ) [pure virtual]
```

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1155) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

#### Parameters

<i>value</i>	- reference to the element to add.
--------------	------------------------------------

#### Returns

true if the element was added

#### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implemented in **decaf::util::AbstractQueue< E >** (p. 165), **decaf::util::PriorityQueue< E >** (p. 2979), **decaf::util::StlList< E >** (p. 3535), **decaf::util::StlSet< E >** (p. 3568), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3535), **decaf::util::StlList< CompositeTask \* >** (p. 3535), **decaf::util::StlList< URI >** (p. 3535), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3535), **decaf::util::StlList< PrimitiveValueNode >** (p. 3535), **decaf::util::StlList< Pointer< Command > >** (p. 3535), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3535), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3535), **decaf::util::StlList< cms::Destination \* >** (p. 3535), **decaf::util::StlList< cms::Session \* >** (p. 3535), **decaf::util::StlList< cms::Connection \* >** (p. 3535), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3568), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3568), **decaf::util::StlSet< Resource \* >** (p. 3568), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3568).

```
6.200.3.2  template<typename E> virtual bool decaf::util::Collection<
            E >::addAll ( const Collection< E > & collection ) throw (
                lang::exceptions::UnsupportedOperationException,
                lang::exceptions::IllegalArgumentException,
                lang::exceptions::IllegalStateException ) [pure virtual]
```

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

#### Parameters

<i>collection</i>	- <b>Collection</b> (p. 1155) whose elements are added to this one.
-------------------	---

#### Returns

true if this collection changed as a result of the call

#### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implemented in **decaf::util::AbstractCollection< E >** (p. 150), **decaf::util::AbstractQueue< E >** (p. 165), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 150), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 150), **decaf::util::AbstractCollection< Resource \* >** (p. 150), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 150), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 150), **decaf::util::AbstractCollection< URI >** (p. 150), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 150), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 150), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 150), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 150), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 150), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 150), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 150), **decaf::util::AbstractCollection< cms::Session \* >** (p. 150), and **decaf::util::AbstractCollection< cms::Connection \* >** (p. 150).

```
6.200.3.3  template<typename E> virtual void decaf::util::Collection< E >::clear ( )
            throw ( lang::exceptions::UnsupportedOperationException ) [pure
                virtual]
```

Removes all of the elements from this collection (optional operation).

This collection will be empty after this method returns unless it throws an exception.

**Exceptions**

<i>UnsupportedOperation</i> <i>Exception</i>
---

Implemented in **decaf::util::AbstractCollection**< **E** > (p. 151), **decaf::util::AbstractQueue**< **E** > (p. 166), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3662), **decaf::util::PriorityQueue**< **E** > (p. 2980), **decaf::util::StlList**< **E** > (p. 3537), **decaf::util::StlSet**< **E** > (p. 3568), **decaf::util::AbstractCollection**< **transport::TransportListener** \* > (p. 151), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 151), **decaf::util::AbstractCollection**< **Resource** \* > (p. 151), **decaf::util::AbstractCollection**< **cms::MessageConsumer** \* > (p. 151), **decaf::util::AbstractCollection**< **CompositeTask** \* > (p. 151), **decaf::util::AbstractCollection**< **URI** > (p. 151), **decaf::util::AbstractCollection**< **ActiveMQSession** \* > (p. 151), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 151), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > > (p. 151), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 151), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 151), **decaf::util::AbstractCollection**< **cms::MessageProducer** \* > (p. 151), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 151), **decaf::util::AbstractCollection**< **cms::Session** \* > (p. 151), **decaf::util::AbstractCollection**< **cms::Connection** \* > (p. 151), **decaf::util::StlList**< **cms::MessageConsumer** \* > (p. 3537), **decaf::util::StlList**< **CompositeTask** \* > (p. 3537), **decaf::util::StlList**< **URI** > (p. 3537), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3537), **decaf::util::StlList**< **PrimitiveValueNode** > > (p. 3537), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3537), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3537), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3537), **decaf::util::StlList**< **cms::Destination** \* > (p. 3537), **decaf::util::StlList**< **cms::Session** \* > (p. 3537), **decaf::util::StlList**< **cms::Connection** \* > (p. 3537), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3568), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3568), **decaf::util::StlSet**< **Resource** \* > (p. 3568), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3568).

6.200.3.4 `template<typename E> virtual bool decaf::util::Collection< E >::contains ( const E & value ) const throw ( lang::Exception ) [pure virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that (*o*==null ? *e*==null : *o.equals(e)*).

**Parameters**

<i>value</i>	- value to check for presence in the collection
--------------	---

**Returns**

true if there is at least one of the elements in the collection

**Exceptions**

<i>Exception</i>
------------------

Implemented in **decaf::util::AbstractCollection**< **E** > (p. 152), **decaf::util::StlList**<



**E** > (p. 3537), **decaf::util::StlSet**< **E** > (p. 3569), **decaf::util::AbstractCollection**< **transport::TransportListener** \* > (p. 152), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 152), **decaf::util::AbstractCollection**< **Resource** \* > (p. 152), **decaf::util::AbstractCollection**< **cms::MessageConsumer** \* > (p. 152), **decaf::util::AbstractCollection**< **CompositeTask** \* > (p. 152), **decaf::util::AbstractCollection**< **URI** > (p. 152), **decaf::util::AbstractCollection**< **ActiveMQSession** \* > (p. 152), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 152), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 152), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 152), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 152), **decaf::util::AbstractCollection**< **cms::MessageProducer** \* > (p. 152), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 152), **decaf::util::AbstractCollection**< **cms::Session** \* > (p. 152), **decaf::util::AbstractCollection**< **cms::Connection** \* > (p. 152), **decaf::util::StlList**< **cms::MessageConsumer** \* > (p. 3537), **decaf::util::StlList**< **CompositeTask** \* > (p. 3537), **decaf::util::StlList**< **URI** > (p. 3537), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3537), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3537), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3537), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3537), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3537), **decaf::util::StlList**< **cms::Destination** \* > (p. 3537), **decaf::util::StlList**< **cms::Session** \* > (p. 3537), **decaf::util::StlList**< **cms::Connection** \* > (p. 3537), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3569), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3569), **decaf::util::StlSet**< **Resource** \* > (p. 3569), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3569).

```
6.200.3.5  template<typename E> virtual bool decaf::util::Collection< E >::containsAll (
            const Collection< E > & collection ) const throw ( lang::Exception ) [pure
            virtual]
```

Returns true if this collection contains all of the elements in the specified collection.

#### Parameters

<i>collection</i>	- <b>Collection</b> (p. 1155) to compare to this one.
-------------------	---

#### Exceptions

<i>Exception</i>
------------------

Implemented in **decaf::util::AbstractCollection**< **E** > (p. 153), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3663), **decaf::util::AbstractCollection**< **transport::TransportListener** \* > (p. 153), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 153), **decaf::util::AbstractCollection**< **Resource** \* > (p. 153), **decaf::util::AbstractCollection**< **cms::MessageConsumer** \* > (p. 153), **decaf::util::AbstractCollection**< **CompositeTask** \* > (p. 153), **decaf::util::AbstractCollection**< **URI** > (p. 153), **decaf::util::AbstractCollection**< **ActiveMQSession** \* > (p. 153), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 153), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 153), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 153), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 153), **decaf::util::AbstractCollection**< **cms::MessageProducer** \* > (p. 153), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 153), **decaf::util::AbstractCollection**< **cms::Session** \* > (p. 153), and **decaf::util::AbstractCollection**< **cms::Connection** \* > (p. 153).

```
6.200.3.6  template<typename E> virtual bool decaf::util::Collection< E >::equals ( const
           Collection< E > & value ) const  [pure virtual]
```

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

### Returns

true if the Collections contain the same elements.

Implemented in `decaf::util::AbstractCollection< E >` (p. 153), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3665), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 153), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 153), `decaf::util::AbstractCollection< Resource * >` (p. 153), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 153), `decaf::util::AbstractCollection< CompositeTask * >` (p. 153), `decaf::util::AbstractCollection< URI >` (p. 153), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 153), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 153), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 153), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 153), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 153), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 153), `decaf::util::AbstractCollection< cms::Destination * >` (p. 153), `decaf::util::AbstractCollection< cms::Session * >` (p. 153), and `decaf::util::AbstractCollection< cms::Connection * >` (p. 153).

```
6.200.3.7  template<typename E> virtual bool decaf::util::Collection< E >::isEmpty ( )
           const  [pure virtual]
```

### Returns

true if this collection contains no elements.

Implemented in `decaf::util::AbstractCollection< E >` (p. 154), `decaf::util::concurrent::SynchronousQueue< E >` (p. 3665), `decaf::util::StlList< E >` (p. 3539), `decaf::util::StlSet< E >` (p. 3570), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 154), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 154), `decaf::util::AbstractCollection< Resource * >` (p. 154), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 154), `decaf::util::AbstractCollection< CompositeTask * >` (p. 154), `decaf::util::AbstractCollection< URI >` (p. 154), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 154), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 154), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 154), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 154), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 154), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 154), `decaf::util::AbstractCollection< cms::Destination * >` (p. 154), `decaf::util::AbstractCollection< cms::Session * >` (p. 154), `decaf::util::AbstractCollection< cms::Connection * >` (p. 154), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3539), `decaf::util::StlList< CompositeTask * >` (p. 3539), `decaf::util::StlList< URI >` (p. 3539), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 3539), `decaf::util::StlList< PrimitiveValueNode >` (p. 3539), `decaf::util::StlList< Pointer< Command > >` (p. 3539), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 3539), `decaf::util::StlList< cms::MessageProducer * >` (p. 3539), `decaf::util::StlList< cms::Destination * >` (p. 3539), `decaf::util::StlList<`

**cms::Session \* >** (p. 3539), **decaf::util::StlList< cms::Connection \* >** (p. 3539), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3570), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3570), **decaf::util::StlSet< Resource \* >** (p. 3570), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3570).

6.200.3.8 `template<typename E> virtual bool decaf::util::Collection< E >::remove ( const E & value ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException ) [pure virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*o*==null ? *e*==null : *o.equals(e)*), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

#### Parameters

<i>value</i>	- reference to the element to remove.
--------------	---------------------------------------

#### Returns

true if the collection was changed

#### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	

Implemented in **decaf::util::AbstractCollection< E >** (p. 156), **decaf::util::PriorityQueue< E >** (p. 2983), **decaf::util::StlList< E >** (p. 3541), **decaf::util::StlSet< E >** (p. 3570), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 156), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 156), **decaf::util::AbstractCollection< Resource \* >** (p. 156), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 156), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 156), **decaf::util::AbstractCollection< URI >** (p. 156), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 156), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 156), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 156), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 156), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 156), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 156), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 156), **decaf::util::AbstractCollection< cms::Session \* >** (p. 156), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 156), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3541), **decaf::util::StlList< CompositeTask \* >** (p. 3541), **decaf::util::StlList< URI >** (p. 3541), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3541), **decaf::util::StlList< PrimitiveValueNode >** (p. 3541), **decaf::util::StlList< Pointer< Command > >** (p. 3541), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3541), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3541), **decaf::util::StlList< cms::Destination \* >** (p. 3541), **decaf::util::StlList< cms::Session \* >** (p. 3541), **decaf::util::StlList< cms::Connection \* >** (p. 3541), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3570), **decaf::util::StlSet<**

**Pointer< Synchronization > >** (p. 3570), **decaf::util::StlSet< Resource \* >** (p. 3570), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3570).

```
6.200.3.9  template<typename E> virtual bool decaf::util::Collection<
            E >::removeAll ( const Collection< E > & collection ) throw
            ( lang::exceptions::UnsupportedOperationException,
              lang::exceptions::IllegalArgumentException ) [pure virtual]
```

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1155) whose elements are to be removed
-------------------	--

#### Returns

true if the collection changed as a result of this call

#### Exceptions

<i>UnsupportedOperation Exception</i>	
<i>IllegalArgumentException</i>	

Implemented in **decaf::util::AbstractCollection< E >** (p. 157), **decaf::util::AbstractSet< E >** (p. 169), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 157), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 157), **decaf::util::AbstractCollection< Resource \* >** (p. 157), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 157), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 157), **decaf::util::AbstractCollection< URI >** (p. 157), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 157), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 157), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 157), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 157), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 157), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 157), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 157), **decaf::util::AbstractCollection< cms::Session \* >** (p. 157), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 157), **decaf::util::AbstractSet< transport::TransportListener \* >** (p. 169), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 169), **decaf::util::AbstractSet< Resource \* >** (p. 169), and **decaf::util::AbstractSet< ActiveMQSession \* >** (p. 169).

```
6.200.3.10  template<typename E> virtual bool decaf::util::Collection<
              E >::retainAll ( const Collection< E > & collection ) throw
              ( lang::exceptions::UnsupportedOperationException,
                lang::exceptions::IllegalArgumentException ) [pure virtual]
```

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1155) whose elements are to be retained
-------------------	---

#### Returns

true if the collection changed as a result of this call

#### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	

Implemented in **decaf::util::AbstractCollection< E >** (p. 157), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 157), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 157), **decaf::util::AbstractCollection< Resource \* >** (p. 157), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 157), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 157), **decaf::util::AbstractCollection< URI >** (p. 157), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 157), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 157), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 157), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 157), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 157), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 157), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 157), **decaf::util::AbstractCollection< cms::Session \* >** (p. 157), and **decaf::util::AbstractCollection< cms::Connection \* >** (p. 157).

```
6.200.3.11  template<typename E> virtual std::size_t decaf::util::Collection< E >::size (
              ) const [pure virtual]
```

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

#### Returns

the number of elements in this collection

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3669), **decaf::util::PriorityQueue<**

**E** > (p. 2983), **decaf::util::StlList**< **E** > (p. 3542), **decaf::util::StlSet**< **E** > (p. 3571), **decaf::util::StlList**< **cms::MessageConsumer** \* > (p. 3542), **decaf::util::StlList**< **CompositeTask** \* > (p. 3542), **decaf::util::StlList**< **URI** > (p. 3542), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 3542), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 3542), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 3542), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 3542), **decaf::util::StlList**< **cms::MessageProducer** \* > (p. 3542), **decaf::util::StlList**< **cms::Destination** \* > (p. 3542), **decaf::util::StlList**< **cms::Session** \* > (p. 3542), **decaf::util::StlList**< **cms::Connection** \* > (p. 3542), **decaf::util::StlSet**< **transport::TransportListener** \* > (p. 3571), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 3571), **decaf::util::StlSet**< **Resource** \* > (p. 3571), and **decaf::util::StlSet**< **ActiveMQSession** \* > (p. 3571).

Referenced by **decaf::util::AbstractCollection**< **cms::Connection** \* >::**equals**(), **decaf::util::AbstractCollection**< **cms::Connection** \* >::**isEmpty**(), **decaf::util::AbstractSet**< **ActiveMQSession** \* >::**removeAll**(), and **decaf::util::AbstractCollection**< **cms::Connection** \* >::**toArray**().

**6.200.3.12** **template**<**typename E**> **virtual std::vector**<**E**> **decaf::util::Collection**< **E** >::**toArray** ( ) **const** [pure virtual]

Returns an array containing all of the elements in this collection.

If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

### Returns

an array of the elements in this collection.

Implemented in **decaf::util::AbstractCollection**< **E** > (p. 158), **decaf::util::concurrent::SynchronousQueue**< **E** > (p. 3670), **decaf::util::AbstractCollection**< **transport::TransportListener** \* > (p. 158), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 158), **decaf::util::AbstractCollection**< **Resource** \* > (p. 158), **decaf::util::AbstractCollection**< **cms::MessageConsumer** \* > (p. 158), **decaf::util::AbstractCollection**< **CompositeTask** \* > (p. 158), **decaf::util::AbstractCollection**< **URI** > (p. 158), **decaf::util::AbstractCollection**< **ActiveMQSession** \* > (p. 158), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 158), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 158), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 158), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 158), **decaf::util::AbstractCollection**< **cms::MessageProducer** \* > (p. 158), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 158), **decaf::util::AbstractCollection**< **cms::Session** \* > (p. 158), and **decaf::util::AbstractCollection**< **cms::Connection** \* > (p. 158).

The documentation for this class was generated from the following file:

- **src/main/decaf/util/Collection.h**

## 6.201 activemq::commands::Command Class Reference

```
#include <src/main/activemq/commands/Command.h>
```

Inheritance diagram for activemq::commands::Command:

## Public Member Functions

- virtual `~Command ()`
- virtual void **setCommandId** (int id)=0  
*Sets the **Command** (p. 1165) Id of this **Message** (p. 2475).*
- virtual int **getCommandId** () const =0  
*Gets the **Command** (p. 1165) Id of this **Message** (p. 2475).*
- virtual void **setResponseRequired** (const bool required)=0  
*Set if this **Message** (p. 2475) requires a **Response** (p. 3227).*
- virtual bool **isResponseRequired** () const =0  
*Is a **Response** (p. 3227) required for this **Command** (p. 1165).*
- virtual std::string **toString** () const =0  
*Returns a provider-specific string that provides information about the contents of the command.*
- virtual `decaf::lang::Pointer< commands::Command > visit (activemq::state::CommandVisitor *visitor)=0` throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- virtual bool **isConnectionInfo** () const =0
- virtual bool **isConsumerInfo** () const =0
- virtual bool **isBrokerInfo** () const =0
- virtual bool **isKeepAliveInfo** () const =0
- virtual bool **isMessage** () const =0
- virtual bool **isMessageAck** () const =0
- virtual bool **isMessageDispatch** () const =0
- virtual bool **isMessageDispatchNotification** () const =0
- virtual bool **isProducerAck** () const =0
- virtual bool **isProducerInfo** () const =0
- virtual bool **isResponse** () const =0
- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

### 6.201.1 Constructor & Destructor Documentation

6.201.1.1 `virtual activemq::commands::Command::~~Command ( ) [inline, virtual]`

### 6.201.2 Member Function Documentation

6.201.2.1 `virtual int activemq::commands::Command::getCommandId ( ) const [pure virtual]`

Gets the **Command** (p. 1165) Id of this **Message** (p. 2475).

#### Returns

**Command** (p. 1165) Id

Implemented in **activemq::commands::BaseCommand** (p. 726).

6.201.2.2 `virtual bool activemq::commands::Command::isBrokerInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 726), and **activemq::commands::BrokerInfo** (p. 860).

6.201.2.3 `virtual bool activemq::commands::Command::isConnectionInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 726), and **activemq::commands::ConnectionInfo** (p. 1328).

6.201.2.4 `virtual bool activemq::commands::Command::isConsumerInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 726), and **activemq::commands::ConsumerInfo** (p. 1431).

6.201.2.5 `virtual bool activemq::commands::Command::isKeepAliveInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 727), and **activemq::commands::KeepAliveInfo** (p. 2227).



6.201.2.6 `virtual bool activemq::commands::Command::isMessage ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 727), and **activemq::commands::Message** (p. 2486).

6.201.2.7 `virtual bool activemq::commands::Command::isMessageAck ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 727), and **activemq::commands::MessageAck** (p. 2524).

6.201.2.8 `virtual bool activemq::commands::Command::isMessageDispatch ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 727), and **activemq::commands::MessageDispatch** (p. 2558).

6.201.2.9 `virtual bool activemq::commands::Command::isMessageDispatchNotification ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 727), and **activemq::commands::MessageDispatchNotification** (p. 2593).

6.201.2.10 `virtual bool activemq::commands::Command::isProducerAck ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 727), and **activemq::commands::ProducerAck** (p. 2986).

6.201.2.11 `virtual bool activemq::commands::Command::isProducerInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 727), and **activemq::commands::ProducerInfo** (p. 3046).

6.201.2.12 `virtual bool activemq::commands::Command::isRemoveInfo ( ) const [pure virtual]`

Implemented in **activemq::commands::BaseCommand** (p. 728), and **activemq::commands::RemoveInfo** (p. 3140).

6.201.2.13 `virtual bool activemq::commands::Command::isRemoveSubscriptionInfo ( ) const` [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 728), and **activemq::commands::RemoveSubscriptionInfo** (p. 3168).

6.201.2.14 `virtual bool activemq::commands::Command::isResponse ( ) const` [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 728), and **activemq::commands::Response** (p. 3230).

6.201.2.15 `virtual bool activemq::commands::Command::isResponseRequired ( ) const` [pure virtual]

Is a **Response** (p. 3227) required for this **Command** (p. 1165).

#### Returns

true if a response is required.

Implemented in **activemq::commands::BaseCommand** (p. 728).

6.201.2.16 `virtual bool activemq::commands::Command::isShutdownInfo ( ) const` [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 728), and **activemq::commands::ShutdownInfo** (p. 3415).

6.201.2.17 `virtual bool activemq::commands::Command::isTransactionInfo ( ) const` [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 728), and **activemq::commands::TransactionInfo** (p. 3788).

6.201.2.18 `virtual bool activemq::commands::Command::isWireFormatInfo ( ) const` [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 729), and **activemq::commands::WireFormatInfo** (p. 3919).

6.201.2.19 `virtual void activemq::commands::Command::setCommandId ( int id )` [pure virtual]

Sets the **Command** (p. 1165) Id of this **Message** (p. 2475).

**Parameters**

<i>id</i>	<b>Command</b> (p. 1165) Id
-----------	-----------------------------

Implemented in **activemq::commands::BaseCommand** (p. 729).

6.201.2.20 `virtual void activemq::commands::Command::setResponseRequired ( const bool  
required ) [pure virtual]`

Set if this **Message** (p. 2475) requires a **Response** (p. 3227).

**Parameters**

<i>required</i>	true if response is required
-----------------	------------------------------

Implemented in **activemq::commands::BaseCommand** (p. 729).

6.201.2.21 `virtual std::string activemq::commands::Command::toString ( ) const [pure  
virtual]`

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 177), **activemq::commands::ActiveMQBytesMessage** (p. 214), **activemq::commands::ActiveMQMapMessage** (p. 344), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 416), **activemq::commands::ActiveMQStreamMessage** (p. 518), **activemq::commands::ActiveMQTextMessage** (p. 635), **activemq::commands::BaseCommand** (p. 729), **activemq::commands::BrokerInfo** (p. 861), **activemq::commands::ConnectionControl** (p. 1241), **activemq::commands::ConnectionError** (p. 1269), **activemq::commands::ConnectionInfo** (p. 1329), **activemq::commands::ConsumerControl** (p. 1372), **activemq::commands::ConsumerInfo** (p. 1432), **activemq::commands::ControlCommand** (p. 1462), **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse** (p. 1552), **activemq::commands::DestinationInfo** (p. 1695), **activemq::commands::ExceptionResponse** (p. 1804), **activemq::commands::FlushCommand** (p. 1902), **activemq::commands::IntegerResponse** (p. 2056), **activemq::commands::KeepAliveInfo** (p. 2228), **activemq::commands::Message** (p. 2490), **activemq::commands::MessageAck** (p. 2525), **activemq::commands::MessageDispatch** (p. 2558), **activemq::commands::MessageDispatchNotification** (p. 2594), **activemq::commands::MessagePull** (p. 2699), **activemq::commands::ProducerAck** (p. 2987), **activemq::commands::ProducerInfo** (p. 3046), **activemq::commands::RemoveInfo** (p. 3140), **activemq::commands::RemoveSubscriptionInfo** (p. 3168), **activemq::commands::ReplayCommand** (p. 3196), **activemq::commands::Response** (p. 3230), **activemq::commands::SessionInfo** (p. 3351), **activemq::commands::ShutdownInfo** (p. 3415), **activemq::commands::TransactionInfo** (p. 3788), and **activemq::commands::WireFormatInfo** (p. 3922).

```
6.201.2.22 virtual decaf::lang::Pointer<commands::Command>
activemq::commands::Command::visit ( activemq::state::CommandVisitor *
visitor ) throw ( exceptions::ActiveMQException ) [pure virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implemented in **activemq::commands::BrokerError** (p. 827), **activemq::commands::BrokerInfo** (p. 861), **activemq::commands::ConnectionControl** (p. 1241), **activemq::commands::ConnectionError** (p. 1269), **activemq::commands::ConnectionInfo** (p. 1329), **activemq::commands::ConsumerControl** (p. 1373), **activemq::commands::ConsumerInfo** (p. 1433), **activemq::commands::ControlCommand** (p. 1462), **activemq::commands::DestinationInfo** (p. 1695), **activemq::commands::FlushCommand** (p. 1903), **activemq::commands::KeepAliveInfo** (p. 2228), **activemq::commands::Message** (p. 2490), **activemq::commands::MessageAck** (p. 2525), **activemq::commands::MessageDispatch** (p. 2558), **activemq::commands::MessageDispatchNotification** (p. 2594), **activemq::commands::MessageFormatInfo** (p. 2699), **activemq::commands::ProducerAck** (p. 2987), **activemq::commands::ProducerInfo** (p. 3047), **activemq::commands::RemoveInfo** (p. 3140), **activemq::commands::RemoveSubscriptionInfo** (p. 3169), **activemq::commands::ReplayCommand** (p. 3196), **activemq::commands::Response** (p. 3230), **activemq::commands::SessionInfo** (p. 3351), **activemq::commands::ShutdownInfo** (p. 3415), **activemq::commands::TransactionInfo** (p. 3788), and **activemq::commands::WireFormatInfo** (p. 3922).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Command.h**

## 6.202 activemq::state::CommandVisitor Class Reference

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

```
#include <src/main/activemq/state/CommandVisitor.h>
```

Inheritance diagram for activemq::state::CommandVisitor:

### Public Member Functions

- virtual **~CommandVisitor** ()
- virtual **decaf::lang::Pointer< commands::Command > processTransactionInfo** (commands::TransactionInfo \*info)=0 throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer< commands::Command > processRemoveInfo** (**commands::RemoveInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConnectionInfo** (**commands::ConnectionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processSessionInfo** (**commands::SessionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processProducerInfo** (**commands::ProducerInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConsumerInfo** (**commands::ConsumerInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConnection** (**commands::ConnectionId** \*id)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSession** (**commands::SessionId** \*id)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveProducer** (**commands::ProducerId** \*id)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (**commands::ConsumerId** \*id)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (**commands::DestinationInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (**commands::DestinationInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (**commands::Message** \*send)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (**commands::MessageAck** \*ack)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (**commands::MessagePull** \*pull)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (**commands::TransactionInfo** \*info)=0 throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionTwoPhase** (commands::TransactionInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRollbackTransaction** (commands::TransactionInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processWireFormat** (commands::WireFormatInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processKeepAliveInfo** (commands::KeepAliveInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processShutdownInfo** (commands::ShutdownInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processFlushCommand** (commands::FlushCommand \*command)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processBrokerInfo** (commands::BrokerInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRecoverTransactions** (commands::TransactionInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processForgetTransaction** (commands::TransactionInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processEndTransaction** (commands::TransactionInfo \*info)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatchNotification** (commands::MessageDispatchNotification \*notification)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processProducerAck** (commands::ProducerAck \*ack)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatch** (commands::MessageDispatch \*dispatch)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processControlCommand** (commands::ControlCommand \*command)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConnectionError** (commands::ConnectionError \*error)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConnectionControl** (commands::ConnectionControl \*control)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConsumerControl** (commands::ConsumerControl \*control)=0 throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** \*error)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** \*replay)=0 throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processResponse** (**commands::Response** \*response)=0 throw ( exceptions::ActiveMQException )

### 6.202.1 Detailed Description

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

The Commands themselves implement a `visit` method that is called with an instance of this interface and each one then call the appropriate `processXXX` method.

Since

3.0

### 6.202.2 Constructor & Destructor Documentation

6.202.2.1 virtual **activemq::state::CommandVisitor**::~**CommandVisitor** ( ) [inline, virtual]

### 6.202.3 Member Function Documentation

6.202.3.1 virtual **decaf::lang::Pointer**<**commands::Command**>  
**activemq::state::CommandVisitor::processBeginTransaction**  
( **commands::TransactionInfo** \* *info* ) throw ( exceptions::ActiveMQException ) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1363).

6.202.3.2 virtual **decaf::lang::Pointer**<**commands::Command**>  
**activemq::state::CommandVisitor::processBrokerError** ( **commands::BrokerError** \* *error* ) throw ( exceptions::ActiveMQException ) [pure virtual]

6.202.3.3 virtual **decaf::lang::Pointer**<**commands::Command**>  
**activemq::state::CommandVisitor::processBrokerInfo** ( **commands::BrokerInfo** \* *info* ) throw ( exceptions::ActiveMQException ) [pure virtual]

6.202.3.4 virtual **decaf::lang::Pointer**<**commands::Command**>  
**activemq::state::CommandVisitor::processCommitTransactionOnePhase**  
( **commands::TransactionInfo** \* *info* ) throw ( exceptions::ActiveMQException ) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1363).

6.202.3.5 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processCommitTransactionTwoPhase`  
`( commands::TransactionInfo * info ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1364).

6.202.3.6 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConnectionControl`  
`( commands::ConnectionControl * control ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

6.202.3.7 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConnectionError`  
`( commands::ConnectionError * error ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

6.202.3.8 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConnectionInfo`  
`( commands::ConnectionInfo * info ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1364).

6.202.3.9 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConsumerControl`  
`( commands::ConsumerControl * control ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

6.202.3.10 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processConsumerInfo`  
`( commands::ConsumerInfo * info ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1364).

6.202.3.11 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processControlCommand`  
`( commands::ControlCommand * command ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

6.202.3.12 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitor::processDestinationInfo`  
`( commands::DestinationInfo * info ) throw (`  
`exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1364).



6.202.3.13 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processEndTransaction  
( commands::TransactionInfo \* *info* ) throw ( exceptions::ActiveMQException ) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1364).

6.202.3.14 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processFlushCommand  
( commands::FlushCommand \* *command* ) throw ( exceptions::ActiveMQException ) [pure virtual]

6.202.3.15 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processForgetTransaction  
( commands::TransactionInfo \* *info* ) throw ( exceptions::ActiveMQException ) [pure virtual]

6.202.3.16 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processKeepAliveInfo  
( commands::KeepAliveInfo \* *info* ) throw ( exceptions::ActiveMQException ) [pure virtual]

6.202.3.17 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessage ( commands::Message \*  
*send* ) throw ( exceptions::ActiveMQException ) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1364).

6.202.3.18 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageAck ( commands::MessageAck \* *ack* ) throw ( exceptions::ActiveMQException ) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1365).

6.202.3.19 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageDispatch  
( commands::MessageDispatch \* *dispatch* ) throw ( exceptions::ActiveMQException ) [pure virtual]

6.202.3.20 virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessageDispatchNotification ( commands::MessageDispatchNotification \* *notification* ) throw ( exceptions::ActiveMQException ) [pure virtual]

6.202.3.21 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processMessagePull (  
commands::MessagePull * pull ) throw ( exceptions::ActiveMQException  
) [pure virtual]`

6.202.3.22 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processPrepareTransaction  
( commands::TransactionInfo * info ) throw (  
exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1365).

6.202.3.23 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processProducerAck (  
commands::ProducerAck * ack ) throw ( exceptions::ActiveMQException  
) [pure virtual]`

6.202.3.24 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processProducerInfo  
( commands::ProducerInfo * info ) throw (  
exceptions::ActiveMQException ) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1365).

6.202.3.25 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRecoverTransactions  
( commands::TransactionInfo * info ) throw (  
exceptions::ActiveMQException ) [pure virtual]`

6.202.3.26 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveConnection (  
commands::ConnectionId * id ) throw ( exceptions::ActiveMQException  
) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1365).

6.202.3.27 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveConsumer (  
commands::ConsumerId * id ) throw ( exceptions::ActiveMQException )  
[pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1365).

6.202.3.28 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveDestination  
( commands::DestinationInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 1365).

6.202.3.29 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveInfo ( commands::RemoveInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in **activemq::state::CommandVisitorAdapter** (p. 1185).

6.202.3.30 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveProducer ( commands::ProducerId * id ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 1366).

6.202.3.31 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveSession ( commands::SessionId * id ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 1366).

6.202.3.32 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processRemoveSubscriptionInfo ( commands::RemoveSubscriptionInfo * info ) throw ( exceptions::ActiveMQException ) [pure virtual]`

6.202.3.33 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processReplayCommand ( commands::ReplayCommand * replay ) throw ( exceptions::ActiveMQException ) [pure virtual]`

6.202.3.34 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitor::processResponse ( commands::Response * response ) throw ( exceptions::ActiveMQException ) [pure virtual]`

```
6.202.3.35 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRollbackTransaction
( commands::TransactionInfo * info ) throw (
exceptions::ActiveMQException ) [pure virtual]
```

Implemented in **activemq::state::ConnectionStateTracker** (p. 1366).

```
6.202.3.36 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processSessionInfo (
commands::SessionInfo * info ) throw ( exceptions::ActiveMQException
) [pure virtual]
```

Implemented in **activemq::state::ConnectionStateTracker** (p. 1366).

```
6.202.3.37 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processShutdownInfo
( commands::ShutdownInfo * info ) throw (
exceptions::ActiveMQException ) [pure virtual]
```

```
6.202.3.38 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processTransactionInfo
( commands::TransactionInfo * info ) throw (
exceptions::ActiveMQException ) [pure virtual]
```

Implemented in **activemq::state::CommandVisitorAdapter** (p. 1186).

```
6.202.3.39 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processWireFormat
( commands::WireFormatInfo * info ) throw (
exceptions::ActiveMQException ) [pure virtual]
```

The documentation for this class was generated from the following file:

- src/main/activemq/state/**CommandVisitor.h**

## 6.203 activemq::state::CommandVisitorAdapter Class Reference

Default Implementation of a **CommandVisitor** (p. 1171) that returns NULL for all calls.

```
#include <src/main/activemq/state/CommandVisitorAdapter.h>
```

Inheritance diagram for **activemq::state::CommandVisitorAdapter**:

## Public Member Functions

- virtual `~CommandVisitorAdapter ()`
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConsumer (commands::ConsumerId *id AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processDestinationInfo (commands::DestinationInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processRemoveDestination (commands::DestinationInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processMessage (commands::Message *send AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processMessageAck (commands::MessageAck *ack AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processMessagePull (commands::MessagePull *pull AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processBeginTransaction (commands::TransactionInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processPrepareTransaction (commands::TransactionInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase (commands::TransactionInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processCommitTransactionTwoPhase (commands::TransactionInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`
- virtual `decaf::lang::Pointer< commands::Command > processRollbackTransaction (commands::TransactionInfo *info AMQCPP_UNUSED) throw ( exceptions::ActiveMQException )`

- virtual **decaf::lang::Pointer**< **commands::Command** > **processWireFormat** (**commands::WireFormatInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processKeepAliveInfo** (**commands::KeepAliveInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processShutdownInfo** (**commands::ShutdownInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processFlushCommand** (**commands::FlushCommand** \*command AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerInfo** (**commands::BrokerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRecoverTransactions** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** \*notification AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** \*ack AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** \*dispatch AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** \*command AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** \*error AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** \*control AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** \*control AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** \*error AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )

- virtual **decaf::lang::Pointer< commands::Command > processReplayCommand** (**commands::ReplayCommand** \*replay AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processResponse** (**commands::Response** \*response AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConnectionInfo** (**commands::ConnectionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processSessionInfo** (**commands::SessionInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processProducerInfo** (**commands::ProducerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processConsumerInfo** (**commands::ConsumerInfo** \*info AMQCPP\_UNUSED) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processTransactionInfo** (**commands::TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **decaf::lang::Pointer< commands::Command > processRemoveInfo** (**commands::RemoveInfo** \*info) throw ( exceptions::ActiveMQException )

### 6.203.1 Detailed Description

Default Implementation of a **CommandVisitor** (p. 1171) that returns NULL for all calls.

Since

3.0

### 6.203.2 Constructor & Destructor Documentation

6.203.2.1 virtual **activemq::state::CommandVisitorAdapter::~CommandVisitorAdapter** ( )  
[inline, virtual]

### 6.203.3 Member Function Documentation

6.203.3.1 virtual **decaf::lang::Pointer< commands::Command >**  
**activemq::state::CommandVisitorAdapter::processBeginTransaction** (**commands::TransactionInfo** \*info *AMQCPP\_UNUSED* ) throw ( exceptions::ActiveMQException ) [inline, virtual]

- 6.203.3.2 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processBrokerError (  
commands::BrokerError *error AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.3 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processBrokerInfo (  
commands::BrokerInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.4 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase  
( commands::TransactionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.5 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase  
( commands::TransactionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.6 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConnectionControl (  
commands::ConnectionControl *control AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.7 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConnectionError (  
commands::ConnectionError *error AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.8 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConnectionInfo (  
commands::ConnectionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.9 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConsumerControl (  
commands::ConsumerControl *control AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`
- 6.203.3.10 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processConsumerInfo (  
commands::ConsumerInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException) [inline, virtual]`



- 6.203.3.11 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processControlCommand (  
commands::ControlCommand *command AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.12 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processDestinationInfo (  
commands::DestinationInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.13 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processEndTransaction (  
commands::TransactionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.14 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processFlushCommand (  
commands::FlushCommand *command AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.15 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processForgetTransaction (  
commands::TransactionInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.16 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processKeepAliveInfo (  
commands::KeepAliveInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.17 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processMessage (  
commands::Message *send AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.18 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processMessageAck (  
commands::MessageAck *ack AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`
- 6.203.3.19 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processMessageDispatch (  
commands::MessageDispatch *dispatch AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`

- 6.203.3.20 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processMessageDispatchNotification (`  
`commands::MessageDispatchNotification *notification AMQCPP_UNUSED )`  
`throw ( exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.21 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processMessagePull (`  
`commands::MessagePull *pull AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.22 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processPrepareTransaction`  
`( commands::TransactionInfo *info AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.23 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processProducerAck (`  
`commands::ProducerAck *ack AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.24 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processProducerInfo (`  
`commands::ProducerInfo *info AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.25 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processRecoverTransactions`  
`( commands::TransactionInfo *info AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.26 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processRemoveConnection`  
`( commands::ConnectionId *id AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.27 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processRemoveConsumer`  
`( commands::ConsumerId *id AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`
  
- 6.203.3.28 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::state::CommandVisitorAdapter::processRemoveDestination`  
`( commands::DestinationInfo *info AMQCPP_UNUSED ) throw (`  
`exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.29 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processRemoveInfo (  
 commands::RemoveInfo * info ) throw ( exceptions::ActiveMQException  
 ) [inline, virtual]`

Implements `activemq::state::CommandVisitor` (p. 1177).

References `activemq::commands::ConnectionId::ID_CONNECTIONID`, `activemq::commands::ConsumerId::ID_CONSUMERID`, `activemq::commands::ProducerId::ID_PRODUCERID`, and `activemq::commands::SessionId::ID_SESSIONID`.

6.203.3.30 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processRemoveProducer  
 ( commands::ProducerId *id AMQCPP_UNUSED ) throw ( exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.31 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processRemoveSession  
 ( commands::SessionId *id AMQCPP_UNUSED ) throw ( exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.32 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo (  
 commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED ) throw ( exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.33 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processReplayCommand (  
 commands::ReplayCommand *replay AMQCPP_UNUSED ) throw ( exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.34 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processResponse (  
 commands::Response *response AMQCPP_UNUSED ) throw ( exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.35 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processRollbackTransaction  
 ( commands::TransactionInfo *info AMQCPP_UNUSED ) throw ( exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.36 `virtual decaf::lang::Pointer<commands::Command>  
 activemq::state::CommandVisitorAdapter::processSessionInfo (  
 commands::SessionInfo *info AMQCPP_UNUSED ) throw ( exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.37 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processShutdownInfo (  
commands::ShutdownInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`

6.203.3.38 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processTransactionInfo  
( commands::TransactionInfo * info ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`

Implements `activemq::state::CommandVisitor` (p. 1178).

References `activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_COMMITONEPHASE`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_COMMITTWOPHASE`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_END`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_FORGET`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_PREPARE`, `activemq::core::ActiveMQConstants::TRANSACTION_STATE_RECOVER`, and `activemq::core::ActiveMQConstants::TRANSACTION_STATE_ROLLBACK`.

6.203.3.39 `virtual decaf::lang::Pointer<commands::Command>  
activemq::state::CommandVisitorAdapter::processWireFormat (  
commands::WireFormatInfo *info AMQCPP_UNUSED ) throw (  
exceptions::ActiveMQException ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/CommandVisitorAdapter.h`

## 6.204 `decaf::lang::Comparable< T >` Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

```
#include <src/main/decaf/lang/Comparable.h>
```

### Public Member Functions

- virtual `~Comparable ()`
- virtual `int compareTo (const T &value) const =0`  
*Compares this object with the specified object for order.*
- virtual `bool equals (const T &value) const =0`
- virtual `bool operator== (const T &value) const =0`  
*Compares equality between this object and the one passed.*
- virtual `bool operator< (const T &value) const =0`  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### 6.204.1 Detailed Description

```
template<typename T>class decaf::lang::Comparable< T >
```

This interface imposes a total ordering on the objects of each class that implements it.

This ordering is referred to as the class's natural ordering, and the class's `compareTo` method is referred to as its natural comparison method.

### 6.204.2 Constructor & Destructor Documentation

```
6.204.2.1 template<typename T> virtual decaf::lang::Comparable< T
>::~~Comparable ( ) [inline, virtual]
```

### 6.204.3 Member Function Documentation

```
6.204.3.1 template<typename T> virtual int decaf::lang::Comparable< T >::compareTo (
const T & value ) const [pure virtual]
```

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all `x` and `y`. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementor must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all `z`.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the **Comparable** (p. 1186) interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

#### Parameters

<i>value</i>	- the Object to be compared.
--------------	------------------------------

#### Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in **decaf::lang::Boolean** (p. 812), **decaf::lang::Boolean** (p. 812), **decaf::lang::Byte** (p. 921), **decaf::lang::Byte** (p. 921), **decaf::lang::Character** (p. 1071), **decaf::lang::Character** (p. 1071), **decaf::lang::Double** (p. 1754), **decaf::lang::Double**

(p. 1754), **decaf::lang::Float** (p. 1868), **decaf::lang::Float** (p. 1868), **decaf::lang::Integer** (p. 2042), **decaf::lang::Integer** (p. 2042), **decaf::lang::Long** (p. 2381), **decaf::lang::Long** (p. 2380), **decaf::lang::Short** (p. 3383), and **decaf::lang::Short** (p. 3383).

6.204.3.2 `template<typename T> virtual bool decaf::lang::Comparable< T >::equals (const T & value ) const` [pure virtual]

### Returns

true if this value is considered equal to the passed value.

Implemented in **decaf::lang::Boolean** (p. 813), **decaf::lang::Boolean** (p. 813), **decaf::lang::Byte** (p. 922), **decaf::lang::Byte** (p. 922), **decaf::lang::Character** (p. 1073), **decaf::lang::Character** (p. 1072), **decaf::lang::Double** (p. 1756), **decaf::lang::Double** (p. 1756), **decaf::lang::Float** (p. 1869), **decaf::lang::Float** (p. 1869), **decaf::lang::Integer** (p. 2043), **decaf::lang::Integer** (p. 2043), **decaf::lang::Long** (p. 2382), **decaf::lang::Long** (p. 2382), **decaf::lang::Short** (p. 3384), and **decaf::lang::Short** (p. 3384).

6.204.3.3 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator< ( const T & value ) const` [pure virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

### Returns

true if this object is equal to the one passed.

Implemented in **decaf::lang::Boolean** (p. 813), **decaf::lang::Boolean** (p. 813), **decaf::lang::Byte** (p. 924), **decaf::lang::Byte** (p. 923), **decaf::lang::Character** (p. 1074), **decaf::lang::Character** (p. 1075), **decaf::lang::Double** (p. 1758), **decaf::lang::Double** (p. 1758), **decaf::lang::Float** (p. 1872), **decaf::lang::Float** (p. 1872), **decaf::lang::Integer** (p. 2046), **decaf::lang::Integer** (p. 2046), **decaf::lang::Long** (p. 2385), **decaf::lang::Long** (p. 2385), **decaf::lang::Short** (p. 3385), and **decaf::lang::Short** (p. 3385).

6.204.3.4 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator== ( const T & value ) const` [pure virtual]

Compares equality between this object and the one passed.

### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

Implemented in **decaf::lang::Boolean** (p. 814), **decaf::lang::Boolean** (p. 814), **decaf::lang::Byte** (p. 924), **decaf::lang::Byte** (p. 924), **decaf::lang::Character** (p. 1075), **decaf::lang::Character** (p. 1075), **decaf::lang::Double** (p. 1758), **decaf::lang::Double** (p. 1759), **decaf::lang::Float** (p. 1873), **decaf::lang::Float** (p. 1873), **decaf::lang::Integer** (p. 2047), **decaf::lang::Integer** (p. 2047), **decaf::lang::Long** (p. 2385), **decaf::lang::Long** (p. 2386), **decaf::lang::Short** (p. 3386), and **decaf::lang::Short** (p. 3386).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Comparable.h

**6.205 decaf::util::Comparator< T > Class Template Reference**

A comparison function, which imposes a total ordering on some collection of objects.

```
#include <src/main/decaf/util/Comparator.h>
```

Inheritance diagram for decaf::util::Comparator< T >:

**Public Member Functions**

- virtual **~Comparator** ()
- virtual bool **operator()** (const T &left, const T &right) const =0  
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1189) to be passed to an STL **Map** (p. 2419) for use as the sorting criteria.*
- virtual int **compare** (const T &o1, const T &o2) const =0  
*Compares its two arguments for order.*

**6.205.1 Detailed Description**

```
template<typename T>class decaf::util::Comparator< T >
```

A comparison function, which imposes a total ordering on some collection of objects.

Comparators can be passed to a sort method (such as Collections.sort) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 1189) c on a set of elements S is said to be consistent with equals if and only if ( compare( e1, e2 ) == 0 ) has the same boolean value as ( e1 == e2 ) for every e1 and e2 in S.

## 6.205.2 Constructor & Destructor Documentation

6.205.2.1 `template<typename T> virtual decaf::util::Comparator< T >::~~Comparator ( ) [inline, virtual]`

## 6.205.3 Member Function Documentation

6.205.3.1 `template<typename T> virtual int decaf::util::Comparator< T >::compare ( const T & o1, const T & o2 ) const [pure virtual]`

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn( compare(x, y)) == -sgn(compare(y, x) )` for all `x` and `y`. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all `z`.

It is generally the case, but not strictly required that `(compare(x, y)==0) == ( x == y )`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

### Parameters

<i>o1</i>	- the first object to be compared
<i>o2</i>	- the second object to be compared

### Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implemented in `decaf::util::comparators::Less< E >` (p. 2288).

6.205.3.2 `template<typename T> virtual bool decaf::util::Comparator< T >::operator() ( const T & left, const T & right ) const [pure virtual]`

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1189) to be passed to an STL **Map** (p. 2419) for use as the sorting criteria.

### Parameters

<i>left</i>	- the Left hand side operand.
<i>right</i>	- the Right hand side operand.



### Returns

true if the vale of left is less than the value of right.

Implemented in **decaf::util::comparators::Less< E >** (p. 2288).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Comparator.h**

## 6.206 activemq::util::CompositeData Class Reference

Represents a Composite URI.

```
#include <src/main/activemq/util/CompositeData.h>
```

### Public Member Functions

- **CompositeData** ()
- virtual **~CompositeData** ()
- **StlList< URI > &getComponents** ()
- const **StlList< URI > &getComponents** () const
- void **setComponents** (const **StlList< URI >** &components)
- std::string **getFragment** () const
- void **setFragment** (const std::string &fragment)
- const **Properties &getParameters** () const
- void **setParameters** (const **Properties** &parameters)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const throw ( decaf::net::URISyntaxException )

### 6.206.1 Detailed Description

Represents a Composite URI.

### Since

3.0

## 6.206.2 Constructor & Destructor Documentation

6.206.2.1 `activemq::util::CompositeData::CompositeData ( )`

6.206.2.2 `virtual activemq::util::CompositeData::~~CompositeData ( )` `[virtual]`

## 6.206.3 Member Function Documentation

6.206.3.1 `StlList<URI>& activemq::util::CompositeData::getComponents ( )` `[inline]`

6.206.3.2 `const StlList<URI>& activemq::util::CompositeData::getComponents ( )` `const` `[inline]`

6.206.3.3 `std::string activemq::util::CompositeData::getFragment ( )` `const` `[inline]`

6.206.3.4 `std::string activemq::util::CompositeData::getHost ( )` `const` `[inline]`

6.206.3.5 `const Properties& activemq::util::CompositeData::getParameters ( )` `const` `[inline]`

6.206.3.6 `std::string activemq::util::CompositeData::getPath ( )` `const` `[inline]`

6.206.3.7 `std::string activemq::util::CompositeData::getScheme ( )` `const` `[inline]`

6.206.3.8 `void activemq::util::CompositeData::setComponents ( const StlList< URI > & components )` `[inline]`

6.206.3.9 `void activemq::util::CompositeData::setFragment ( const std::string & fragment )` `[inline]`

6.206.3.10 `void activemq::util::CompositeData::setHost ( const std::string & host )` `[inline]`

6.206.3.11 `void activemq::util::CompositeData::setParameters ( const Properties & parameters )` `[inline]`

6.206.3.12 `void activemq::util::CompositeData::setPath ( const std::string & path )` `[inline]`

6.206.3.13 `void activemq::util::CompositeData::setScheme ( const std::string & scheme )` `[inline]`

6.206.3.14 `URI activemq::util::CompositeData::toURI ( )` `const` `throw ( decaf::net::URISyntaxException )`

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CompositeData.h`

## 6.207 activemq::threads::CompositeTask Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p.1194).

```
#include <src/main/activemq/threads/CompositeTask.h>
```

Inheritance diagram for activemq::threads::CompositeTask:

### Public Member Functions

- virtual **~CompositeTask** ()
- virtual bool **isPending** () const =0

*Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3678) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.*

#### 6.207.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p.1194).

#### Since

3.0

#### 6.207.2 Constructor & Destructor Documentation

6.207.2.1 virtual activemq::threads::CompositeTask::~CompositeTask ( ) [inline, virtual]

#### 6.207.3 Member Function Documentation

6.207.3.1 virtual bool activemq::threads::CompositeTask::isPending ( ) const [pure virtual]

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.3678) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.

#### Since

3.0

Implemented in `activemq::transport::failover::BackupTransportPool` (p. 722), `activemq::transport::failover::CloseTransportsTask` (p. 1122), and `activemq::transport::failover::FailoverTransport` (p. 1841).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTask.h`

## 6.208 `activemq::threads::CompositeTaskRunner` Class Reference

A **Task** (p. 3678) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.

```
#include <src/main/activemq/threads/CompositeTaskRunner.h>
```

Inheritance diagram for `activemq::threads::CompositeTaskRunner`:

### Public Member Functions

- **CompositeTaskRunner** ()
- virtual **~CompositeTaskRunner** ()
- void **addTask** (**CompositeTask** \*task)  
*Adds a new **CompositeTask** (p. 1193) to the Set of Tasks that this class manages.*
- void **removeTask** (**CompositeTask** \*task)  
*Removes a **CompositeTask** (p. 1193) that was added previously.*
- virtual void **shutdown** (unsigned int timeout)  
*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()  
*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()  
*Signal the **TaskRunner** (p. 3680) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3678) instance will be run until its iterate method has returned false indicating it is done.*

### Protected Member Functions

- virtual void **run** ()  
*Run method - called by the Thread class in the context of the thread.*
- virtual bool **iterate** ()  
*Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.*

### 6.208.1 Detailed Description

A **Task** (p. 3678) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

#### Since

3.0

### 6.208.2 Constructor & Destructor Documentation

6.208.2.1 `activemq::threads::CompositeTaskRunner::CompositeTaskRunner ( )`

6.208.2.2 `virtual activemq::threads::CompositeTaskRunner::~~CompositeTaskRunner ( )`  
[virtual]

### 6.208.3 Member Function Documentation

6.208.3.1 `void activemq::threads::CompositeTaskRunner::addTask ( CompositeTask * task )`

Adds a new **CompositeTask** (p. 1193) to the Set of Tasks that this class manages.

#### Parameters

<i>task</i>	- Pointer to a <b>CompositeTask</b> (p. 1193) instance.
-------------	---

6.208.3.2 `virtual bool activemq::threads::CompositeTaskRunner::iterate ( )`  
[protected, virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

#### Returns

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implements **activemq::threads::Task** (p. 3679).

6.208.3.3 `void activemq::threads::CompositeTaskRunner::removeTask ( CompositeTask * task )`

Removes a **CompositeTask** (p. 1193) that was added previously.

#### Parameters

<i>task</i>	- Pointer to a <b>CompositeTask</b> (p. 1193) instance.
-------------	---

6.208.3.4 `virtual void activemq::threads::CompositeTaskRunner::run ( )` [protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3265).

6.208.3.5 `virtual void activemq::threads::CompositeTaskRunner::shutdown ( )` [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3681).

6.208.3.6 `virtual void activemq::threads::CompositeTaskRunner::shutdown ( unsigned int timeout )` [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

#### Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implements **activemq::threads::TaskRunner** (p. 3681).

6.208.3.7 `virtual void activemq::threads::CompositeTaskRunner::wakeup ( )` [virtual]

Signal the **TaskRunner** (p. 3680) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3678) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3681).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTaskRunner.h`

## 6.209 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 3819) is a **Transport** (p. 3819) implementation that is composed of several Transports.

```
#include <src/main/activemq/transport/CompositeTransport.h>
```

Inheritance diagram for `activemq::transport::CompositeTransport`:

## Public Member Functions

- virtual `~CompositeTransport ()`
- virtual void `addURI (const List< URI > &uris)=0`  
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3819) is a composite of.*
- virtual void `removeURI (const List< URI > &uris)=0`  
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3819) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3819) should result in that **Transport** (p. 3819) being disposed of.*

### 6.209.1 Detailed Description

A Composite **Transport** (p. 3819) is a **Transport** (p. 3819) implementation that is composed of several Transports.

The composition could be such that only one **Transport** (p. 3819) exists for each URI that is composed or there could be many active Transports working at once.

#### Since

3.0

### 6.209.2 Constructor & Destructor Documentation

6.209.2.1 `virtual activemq::transport::CompositeTransport::~CompositeTransport ( )`  
`[inline, virtual]`

### 6.209.3 Member Function Documentation

6.209.3.1 `virtual void activemq::transport::CompositeTransport::addURI ( const List< URI > &uris )` `[pure virtual]`

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3819) is a composite of.

#### Parameters

<i>uris</i>	The new URI set to add to the set this composite maintains.
-------------	---

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1838).

6.209.3.2 `virtual void activemq::transport::CompositeTransport::removeURI ( const List< URI > &uris )` `[pure virtual]`

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3819) is composed of, removing a URI for which the composite has created a

connected **Transport** (p. 3819) should result in that **Transport** (p. 3819) being disposed of.

#### Parameters

<i>uris</i>	The new URI set to remove to the set this composite maintains.
-------------	--

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1842).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**CompositeTransport.h**

## 6.210 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 2419) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2419) interface.

```
#include <src/main/decaf/util/concurrent/ConcurrentMap.h>
```

Inheritance diagram for decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >:

### Public Member Functions

- virtual **~ConcurrentMap** ()
- virtual bool **putIfAbsent** (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*If the specified key is not already associated with a value, associate it with the given value.*
- virtual bool **remove** (const K &key, const V &value)=0  
*Remove entry for key only if currently mapped to given value.*
- virtual bool **replace** (const K &key, const V &oldValue, const V &newValue)=0  
*Replace entry for key only if currently mapped to given value.*
- virtual V **replace** (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Replace entry for key only if currently mapped to some value.*

#### 6.210.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR>class decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >
```

Interface for a **Map** (p. 2419) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2419) interface.



Since

1.0

## 6.210.2 Constructor & Destructor Documentation

6.210.2.1 `template<typename K, typename V, typename COMPARATOR> virtual  
 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR  
 >::~~ConcurrentMap( ) [inline, virtual]`

## 6.210.3 Member Function Documentation

6.210.3.1 `template<typename K, typename V, typename COMPARATOR>  
 virtual bool decaf::util::concurrent::ConcurrentMap< K, V,  
 COMPARATOR >::putIfAbsent ( const K & key, const V & value ) throw (   
 decaf::lang::exceptions::UnsupportedOperationException ) [pure  
 virtual]`

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

### Parameters

<i>key</i>	The key to map the value to.
<i>value</i>	The value to map to the given key.

### Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

### Exceptions

<i>UnsupportedOperationException</i>	if the put operation is not supported by this map
--------------------------------------	---

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
 > (p. 1214)`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >,  
 Pointer< Message >, MessageId::COMPARATOR > (p. 1214)`, `decaf::util::concurrent::ConcurrentStlMap<`

**Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR**  
**> (p. 1214), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId**  
**>, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1214), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >**  
**(p. 1214), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId**  
**>, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1214),**  
**and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<**  
**ProducerState >, ProducerId::COMPARATOR > (p. 1214).**

**6.210.3.2** `template<typename K, typename V, typename COMPARATOR> virtual bool`  
`decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::remove (`  
`const K & key, const V & value ) [pure virtual]`

Remove entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

#### Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value associated with the specified key.

#### Returns

true if the value was removed, false otherwise

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR**  
**> (p. 1214), decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >,**  
**Pointer< Message >, Messageld::COMPARATOR > (p. 1214), decaf::util::concurrent::ConcurrentStlMap<**  
**Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR**  
**> (p. 1214), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId**  
**>, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1214), decaf::util::concurrent::ConcurrentStlMap<**  
**Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >**  
**(p. 1214), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId**  
**>, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1214),**  
**and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<**  
**ProducerState >, ProducerId::COMPARATOR > (p. 1214).**

## 6.210 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference 1205

6.210.3.3 `template<typename K, typename V, typename COMPARATOR> virtual bool  
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (  
const K & key, const V & oldValue, const V & newValue ) [pure virtual]`

Replace entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) ) {  
    map.put( key, newValue );  
    return true;  
} else {  
    return false;  
}
```

except that the action is performed atomically.

### Parameters

<i>key</i>	key with which the specified value is associated.
<i>oldValue</i>	value expected to be associated with the specified key.
<i>newValue</i>	value to be associated with the specified key.

### Returns

true if the value was replaced

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1216), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >` (p. 1216), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1216), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1216), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1216), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1216), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1216).

6.210.3.4 `template<typename K, typename V, typename COMPARATOR>  
virtual V decaf::util::concurrent::ConcurrentMap< K, V,  
COMPARATOR >::replace ( const K & key, const V & value ) throw (  
decaf::lang::exceptions::NoSuchElementException ) [pure  
virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( ( map.containsKey( key ) ) ) {
```

```

        return map.put( key, value );
    } else {
        throw NoSuchElementException(...);
    };

```

except that the action is performed atomically.

#### Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value to be associated with the specified key.

#### Returns

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

#### Exceptions

<i>NoSuchElementException</i>	if there was no previous mapping.
-------------------------------	-----------------------------------

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1215), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1215).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ConcurrentMap.h`

### 6.211 `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` Class Template Reference

**Map** (p. 2419) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

```
#include <src/main/decaf/util/concurrent/ConcurrentStlMap.h>
```

Inheritance diagram for `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >`:

## Public Member Functions

- **ConcurrentStlMap** ()  
*Default constructor - does nothing.*
- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)  
*Copy constructor - copies the content of the given map into this one.*
- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)  
*Copy constructor - copies the content of the given map into this one.*
- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const
  
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const  
*Comparison, equality is dependent on the method of determining if the element are equal.*
- virtual void **copy** (const **ConcurrentStlMap** &source)
  
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)  
*Copies the content of the source map into this map.*
- virtual void **clear** () throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*

### Exceptions

UnsupportedOperationException	if this map is unmodifiable.
-------------------------------	------------------------------

- virtual bool **containsKey** (const K &key) const  
*Indicates whether or this map contains a value for the given key.*

### Parameters

key	The key to look up.
-----	---------------------

### Returns

*true if this map contains the value, otherwise false.*

- virtual bool **containsValue** (const V &value) const  
*Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.*

### Parameters

value	The Value to look up.
-------	-----------------------

### Returns

*true if this map contains the value, otherwise false.*

- virtual bool **isEmpty** () const

### Returns

*if the **Map** (p. 2419) contains any element or not, TRUE or FALSE*

- virtual std::size\_t **size** () const

**Returns**

*The number of elements (key/value pairs) in this map.*

- virtual V & **get** (const K &key) throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2419).*

*If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.*

**Parameters**

key	<i>The search key.</i>
-----	------------------------

**Returns**

*A reference to the value for the given key.*

**Exceptions**

NoSuchElementException	<i>if the key requests doesn't exist in the <b>Map</b> (p. 2419).</i>
------------------------	---

- virtual const V & **get** (const K &key) const throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2419).*

*If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.*

**Parameters**

key	<i>The search key.</i>
-----	------------------------

**Returns**

*A {const} reference to the value for the given key.*

**Exceptions**

NoSuchElementException	<i>if the key requests doesn't exist in the <b>Map</b> (p. 2419).</i>
------------------------	---

- virtual void **put** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Sets the value for the specified key.*

**Parameters**

key	<i>The target key.</i>
value	<i>The value to be set.</i>

**Exceptions**

UnsupportedOperationException	<i>if this map is unmodifiable.</i>
-------------------------------	-------------------------------------

- virtual void **putAll** (const **ConcurrentStlMap**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Stores a copy of the Mappings contained in the other **Map** (p. 2419) in this one.*

## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 1209

### Parameters

other	A <b>Map</b> (p. 2419) instance whose elements are to all be inserted in this <b>Map</b> (p. 2419).
-------	---

### Exceptions

UnsupportedOperationExcep- tionException	If the implementing class does not support the putAll operation.
---	--

- virtual V **remove** (const K &key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

*Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.*

### Parameters

key	The search key.
-----	-----------------

### Returns

*a copy of the element that was previously mapped to the given key*

### Exceptions

NoSuchElementExcep- tion	if this key is not in the <b>Map</b> (p. 2419).
UnsupportedOperationExcep- tionException	if this map is unmodifiable.

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 3379) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2115), **Set.remove** (p. 156), removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

### Returns

*the entire set of keys in this map as a std::vector.*

- virtual std::vector< V > **values** () const

### Returns

*the entire set of values in this map as a std::vector.*

- bool **putIfAbsent** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*If the specified key is not already associated with a value, associate it with the given value.*

- bool **remove** (const K &key, const V &value)

*Remove entry for key only if currently mapped to given value.*

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

*Replace entry for key only if currently mapped to given value.*

- V **replace** (const K &key, const V &value) throw ( decaf::lang::exceptions::NoSuchElementException )

*Replace entry for key only if currently mapped to some value.*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to **Lock** (p. 2334) the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.211.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >
```

**Map** (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

This version of **Map** (p. 2419) extends the **ConcurrentMap** (p. 1198) interface and implements all the methods defined in that interface. Unlike a Java ConcurrentHashMap this implementation synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java HashTable.

**Since**

1.0

### 6.211.2 Constructor & Destructor Documentation

```
6.211.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::ConcurrentStlMap ( ) [inline]
```

Default constructor - does nothing.



## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 1211

6.211.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
>::ConcurrentStlMap ( const ConcurrentStlMap< K, V, COMPARATOR > &  
source ) [inline]`

Copy constructor - copies the content of the given map into this one.

### Parameters

<i>source</i>	The source map.
---------------	-----------------

6.211.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
>::ConcurrentStlMap ( const Map< K, V, COMPARATOR > & source )  
[inline]`

Copy constructor - copies the content of the given map into this one.

### Parameters

<i>source</i>	The source map.
---------------	-----------------

6.211.2.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR  
>::~~ConcurrentStlMap ( ) [inline, virtual]`

## 6.211.3 Member Function Documentation

6.211.3.1 `template<typename K, typename V, typename  
COMPARATOR = std::less<K>> virtual void  
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::clear (  
) throw ( decaf::lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Removes all keys and values from this map.

### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2421).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

```
6.211.3.2  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::containsKey ( const K & key ) const    [inline, virtual]
```

Indicates whether or this map contains a value for the given key.

#### Parameters

<i>key</i>	The key to look up.
------------	---------------------

#### Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2421).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::remove(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace().

```
6.211.3.3  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::containsValue ( const V & value ) const    [inline, virtual]
```

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

#### Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

#### Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2422).

```
6.211.3.4  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::copy ( const ConcurrentStlMap< K, V, COMPARATOR > & source )
            [inline, virtual]
```

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap().

## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 1213

```
6.211.3.5  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::copy ( const Map< K, V, COMPARATOR > & source ) [inline,
            virtual]
```

Copies the content of the source map into this map.

Erases all existing data in this map.

### Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2423).

```
6.211.3.6  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::equals ( const Map< K, V, COMPARATOR > & source ) const [inline,
            virtual]
```

Comparison, equality is dependent on the method of determining if the element are equal.

### Parameters

<i>source</i>	- <b>Map</b> (p. 2419) to compare to this one.
---------------	--

### Returns

true if the **Map** (p. 2419) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2423).

```
6.211.3.7  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::equals ( const ConcurrentStlMap< K, V, COMPARATOR > & source ) const
            [inline, virtual]
```

```
6.211.3.8  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::get ( const K & key ) throw ( lang::exceptions::NoSuchElementException
            ) [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2419).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

### Parameters

<i>key</i>	The search key.
------------	-----------------

**Returns**

A reference to the value for the given key.

**Exceptions**

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2419).
-------------------------------	--

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2424).

```
6.211.3.9  template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual const V& decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARTOR >::get ( const K & key ) const throw (
lang::exceptions::NoSuchElementException ) [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2419).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

**Parameters**

<i>key</i>	The search key.
------------	-----------------

**Returns**

A {const} reference to the value for the given key.

**Exceptions**

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2419).
-------------------------------	--

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2425).

```
6.211.3.10 template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARTOR
>::isEmpty ( ) const [inline, virtual]
```

**Returns**

if the **Map** (p. 2419) contains any element or not, **TRUE** or **FALSE**

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2426).

```
6.211.3.11 template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual std::vector<K> decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARTOR >::keySet ( ) const [inline, virtual]
```

Returns a **Set** (p. 3379) view of the mappings contained in this map.

## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 1215

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2115), **Set.remove** (p. 156), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

### Returns

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2426).

```
6.211.3.12  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::lock ( ) throw ( decaf::lang::exceptions::RuntimeException )
            [inline, virtual]
```

Locks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

```
6.211.3.13  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::notify ( ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
            virtual]
```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

```
6.211.3.14  template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARTOR
>::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

```
6.211.3.15  template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K,
V, COMPARTOR >::put ( const K & key, const V & value ) throw (
decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Sets the value for the specified key.

### Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implements **decaf::util::Map< K, V, COMPARTOR >** (p. 2427).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARTOR >::putAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARTOR >::putIfAbsent()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARTOR >::replace()`.

## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 1217

```
6.211.3.16  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::putAll ( const Map< K, V, COMPARATOR > & other )
throw ( decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Stores a copy of the Mappings contained in the other **Map** (p. 2419) in this one.

### Parameters

<i>other</i>	A <b>Map</b> (p. 2419) instance whose elements are to all be inserted in this <b>Map</b> (p. 2419).
--------------	---

### Exceptions

<i>UnsupportedOperationException</i>	If the implementing class does not support the putAll operation.
--------------------------------------	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2428).

```
6.211.3.17  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::putAll ( const ConcurrentStlMap< K, V, COMPARATOR > & other )
throw ( decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Referenced by **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()**.

```
6.211.3.18  template<typename K, typename V, typename COMPARATOR =
std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARATOR >::putIfAbsent ( const K & key, const V & value )
throw ( decaf::lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

**Parameters**

<i>key</i>	The key to map the value to.
<i>value</i>	The value to map to the given key.

**Returns**

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

**Exceptions**

<i>UnsupportedOperationException</i>	if the put operation is not supported by this map
--------------------------------------	---

Implements **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR > (p. 1199).

```
6.211.3.19  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::remove( const K & key, const V & value ) [inline, virtual]
```

Remove entry for key only if currently mapped to given value.

**Acts as**

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

**Parameters**

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value associated with the specified key.

**Returns**

true if the value was removed, false otherwise

Implements **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR > (p. 1200).



## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 1219

6.211.3.20 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove ( const K & key ) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

### Parameters

<i>key</i>	The search key.
------------	-----------------

### Returns

a copy of the element that was previously mapped to the given key

### Exceptions

<i>NoSuchElementException</i>	if this key is not in the <b>Map</b> (p. 2419).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2429).

6.211.3.21 `template<typename K, typename V, typename COMPARATOR = std::less<K>> V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace ( const K & key, const V & value ) throw ( decaf::lang::exceptions::NoSuchElementException ) [inline, virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

### Parameters

<i>key</i>	key with which the specified value is associated.
<i>value</i>	value to be associated with the specified key.

**Returns**

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

**Exceptions**

<i>NoSuchElementException</i>	if there was no previous mapping.
-------------------------------	-----------------------------------

Implements **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR > (p. 1202).

```
6.211.3.22  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::replace ( const K & key, const V & oldValue, const V & newValue )
             [inline, virtual]
```

Replace entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

**Parameters**

<i>key</i>	key with which the specified value is associated.
<i>oldValue</i>	value expected to be associated with the specified key.
<i>newValue</i>	value to be associated with the specified key.

**Returns**

true if the value was replaced

Implements **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR > (p. 1201).

```
6.211.3.23  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual std::size_t decaf::util::concurrent::ConcurrentStlMap< K, V,
             COMPARATOR >::size ( ) const [inline, virtual]
```

**Returns**

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map**< K, V, COMPARATOR > (p. 2430).

## 6.211 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference 1221

```
6.211.3.24  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException )
            [inline, virtual]
```

Attempts to **Lock** (p. 2334) the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

```
6.211.3.25  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
            >::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException )
            [inline, virtual]
```

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

```
6.211.3.26  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::vector<V> decaf::util::concurrent::ConcurrentStlMap< K, V,
            COMPARATOR >::values ( ) const [inline, virtual]
```

### Returns

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2430).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values().

```

6.211.3.27  template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARTOR >::wait ( long long millisecs ) throw
( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

```

6.211.3.28  template<typename K, typename V, typename COMPARTOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARTOR
>::wait ( ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

```

6.211.3.29  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARATOR >::wait ( long long millisecs, int nanos
) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentStlMap.h**

## 6.212 decaf::util::concurrent::locks::Condition Class Reference

**Condition** (p. 1220) factors out the **Mutex** (p. 2736) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2336) implementations.

```
#include <src/main/decaf/util/concurrent/locks/Condition.h>
```

## Public Member Functions

- virtual `~Condition()`
- virtual void **await** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted.*
- virtual void **awaitUninterruptibly** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled.*
- virtual long long **awaitNanos** (long long nanosTimeout)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.*
- virtual bool **await** (long long time, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.*
- virtual bool **awaitUntil** (const **Date** &deadline)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException )
- virtual void **signal** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Wakes up one waiting thread.*
- virtual void **signalAll** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Wakes up all waiting threads.*

### 6.212.1 Detailed Description

**Condition** (p. 1220) factors out the **Mutex** (p. 2736) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2336) implementations.

Where a **Lock** (p. 2336) replaces the use of synchronized statements, a **Condition** (p. 1220) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs in different threads, it must be protected, so a lock of some form is associated with the condition. The key property that waiting for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 1220) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 1220) instance for a particular **Lock** (p. 2336) instance use its `newCondition()` method.

As an example, suppose we have a bounded buffer which supports put and take methods. If a take is attempted on an empty buffer, then the thread will block until an item

becomes available; if a put is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting put threads and take threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 1220) instances.

```
class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();
```

```
Object* items = new Object[100]; int putptr, takeptr, count;
```

```
public void put( Object* x ) throw( InterruptedException ) { lock->lock(); try { while( count == 100 ) notFull->await() (p. 1222); items[putptr] = x; if (++putptr == 100) putptr = 0; ++count; notEmpty->signal() (p. 1226); } catch(...) { lock->unlock(); } }
```

```
public Object take() throw( InterruptedException ) { lock->lock(); try { while(count == 0) notEmpty->await() (p. 1222); Object x = items[takeptr]; if (++takeptr == 100) takeptr = 0; --count; notFull->signal() (p. 1226); return x; } catch(...) { lock->unlock(); } }
```

(The `ArrayBlockingQueue` class provides this functionality, so there is no reason to implement this sample usage class.)

#### Implementation Considerations

When waiting upon a **Condition** (p. 1220), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most application programs as a **Condition** (p. 1220) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

#### Since

1.0

## 6.212.2 Constructor & Destructor Documentation

6.212.2.1 `virtual decaf::util::concurrent::locks::Condition::~~Condition ( ) [inline, virtual]`

## 6.212.3 Member Function Documentation

6.212.3.1 `virtual void decaf::util::concurrent::locks::Condition::await ( )  
throw ( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [pure  
virtual]`

Causes the current thread to wait until it is signaled or interrupted.

The lock associated with this **Condition** (p. 1220) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

- \* Some other thread invokes the **signal()** (p. 1226) method for this **Condition** (p. 1220) and the current thread happens to be chosen as the thread to be awakened; or
- \* Some other thread invokes the **signalAll()** (p. 1226) method for this **Condition** (p. 1220); or
- \* Some other thread interrupts the current thread, and interruption of thread suspension is supported; or
- \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

- \* has its interrupted status set on entry to this method; or
- \* is interrupted while waiting and interruption of thread suspension is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1220) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1220).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)



<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.
-------------------------------------	--------------------------------------

6.212.3.2 `virtual bool decaf::util::concurrent::locks::Condition::await ( long long time, const TimeUnit & unit ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

This method is behaviorally equivalent to:

`awaitNanos(unit.toNanos(time)) > 0`

#### Parameters

<i>time</i>	- the maximum time to wait
<i>unit</i>	- the time unit of the time argument

#### Returns

false if the waiting time detectably elapsed before return from the method, else true

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1220).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

6.212.3.3 `virtual long long decaf::util::concurrent::locks::Condition::awaitNanos ( long long nanosTimeout ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

\* Some other thread invokes the **signal()** (p. 1226) method for this **Condition** (p. 1220) and the current thread happens to be chosen as the thread to be awakened; or \* Some

other thread invokes the **signalAll()** (p. 1226) method for this **Condition** (p. 1220); or \* Some other thread interrupts the current thread, and interruption of thread suspension is supported; or \* The specified waiting time elapses; or \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting and interruption of thread suspension is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied nanosTimeout value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

```
synchronized boolean aMethod( long timeout, const TimeUnit& unit ) { long nanosTime-
out = unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTimeout > 0)
nanosTimeout = theCondition->awaitNanos(nanosTimeout); else return false; } // ... }
```

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

#### Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1220) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as IllegalMonitorStateException) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

#### Parameters

<i>nanosTime- out</i>	- the maximum time to wait, in nanoseconds
---------------------------	--

#### Returns

an estimate of the nanosTimeout value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

**Exceptions**

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1220).
<i>InterruptedException</i>	if the current thread is interrupted (and interruption of thread suspension is supported)
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

```
6.212.3.4 virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly
( ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException ) [pure
virtual]
```

Causes the current thread to wait until it is signalled.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes the **signal()** (p. 1226) method for this **Condition** (p. 1220) and the current thread happens to be chosen as the thread to be awakened; or \* Some other thread invokes the **signalAll()** (p. 1226) method for this **Condition** (p. 1220); or \* A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

**Implementation Considerations**

The current thread is assumed to hold the lock associated with this **Condition** (p. 1220) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as *IllegalMonitorStateException*) and the implementation must document that fact.

**Exceptions**

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1220).
<i>IllegalMonitorStateException</i>	if the caller is not the lock owner.

- 6.212.3.5 `virtual bool decaf::util::concurrent::locks::Condition::awaitUntil ( const Date & deadline ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`
- 6.212.3.6 `virtual void decaf::util::concurrent::locks::Condition::signal ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Wakes up one waiting thread.

If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1220).
-------------------------	---

- 6.212.3.7 `virtual void decaf::util::concurrent::locks::Condition::signalAll ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Wakes up all waiting threads.

If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while trying to wait on the <b>Condition</b> (p. 1220).
-------------------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Condition.h`

## 6.213 decaf::util::concurrent::ConditionHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h>
```

#### Public Member Functions

- `ConditionHandle ()`
- `~ConditionHandle ()`
- `ConditionHandle ()`
- `~ConditionHandle ()`

## Data Fields

- pthread\_cond\_t **condition**
- **MutexHandle** \* **mutex**
- HANDLE **semaphore**
- CRITICAL\_SECTION **criticalSection**
- volatile unsigned int **numWaiting**
- volatile unsigned int **numWake**
- volatile unsigned int **generation**

## 6.213.1 Constructor & Destructor Documentation

6.213.1.1 decaf::util::concurrent::ConditionHandle::ConditionHandle ( ) [inline]

6.213.1.2 decaf::util::concurrent::ConditionHandle::~~ConditionHandle ( ) [inline]

6.213.1.3 decaf::util::concurrent::ConditionHandle::ConditionHandle ( ) [inline]

6.213.1.4 decaf::util::concurrent::ConditionHandle::~~ConditionHandle ( ) [inline]

## 6.213.2 Field Documentation

6.213.2.1 pthread\_cond\_t decaf::util::concurrent::ConditionHandle::condition

6.213.2.2 CRITICAL\_SECTION decaf::util::concurrent::ConditionHandle::criticalSection

6.213.2.3 volatile unsigned int decaf::util::concurrent::ConditionHandle::generation

6.213.2.4 **MutexHandle** \* decaf::util::concurrent::ConditionHandle::mutex

6.213.2.5 volatile unsigned int decaf::util::concurrent::ConditionHandle::numWaiting

6.213.2.6 volatile unsigned int decaf::util::concurrent::ConditionHandle::numWake

6.213.2.7 HANDLE decaf::util::concurrent::ConditionHandle::semaphore

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/**ConditionHandle.h**
- src/main/decaf/internal/util/concurrent/windows/**ConditionHandle.h**

## 6.214 decaf::internal::util::concurrent::ConditionImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/ConditionImpl.h>
```

## Static Public Member Functions

- static **decaf::util::concurrent::ConditionHandle \* create** (**decaf::util::concurrent::MutexHandle \*mutex**)  
*Creates the Condition object and attaches it to the given MutexHandle.*
- static void **destroy** (**decaf::util::concurrent::ConditionHandle \*handle**)  
*Destroy a previously create Condition instance.*
- static void **wait** (**decaf::util::concurrent::ConditionHandle \*condition**)  
*Waits for the condition to be signaled.*
- static void **wait** (**decaf::util::concurrent::ConditionHandle \*condition**, long long mills, long long nanos)  
*Waits for the condition to be signaled or for the time specified to ellapse.*
- static void **notify** (**decaf::util::concurrent::ConditionHandle \*condition**)  
*Signals one Thread that is waiting on this condition to wake up.*
- static void **notifyAll** (**decaf::util::concurrent::ConditionHandle \*condition**)  
*Signals all Threads that is waiting on this condition to wake up.*

### 6.214.1 Member Function Documentation

- 6.214.1.1 **static decaf::util::concurrent::ConditionHandle\* decaf::internal::util::concurrent::ConditionImpl::create ( decaf::util::concurrent::MutexHandle \* mutex )** [static]

Creates the Condition object and attaches it to the given MutexHandle.

#### Parameters

<i>mutex</i>	the Mutex handle that this Condition is attached to.
--------------	--

#### Returns

a newly constructed Condition handle that is attached to the given handle.

- 6.214.1.2 **static void decaf::internal::util::concurrent::ConditionImpl::destroy ( decaf::util::concurrent::ConditionHandle \* handle )** [static]

Destroy a previously create Condition instance.

#### Parameters

<i>handle</i>	The Condition handle to be destroyed.
---------------	---------------------------------------

6.214.1.3 static void decaf::internal::util::concurrent::ConditionImpl::notify ( decaf::util::concurrent::ConditionHandle \* *condition* ) [static]

Signals one Thread that is waiting on this condition to wake up.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

6.214.1.4 static void decaf::internal::util::concurrent::ConditionImpl::notifyAll ( decaf::util::concurrent::ConditionHandle \* *condition* ) [static]

Signals all Threads that is waiting on this condition to wake up.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

6.214.1.5 static void decaf::internal::util::concurrent::ConditionImpl::wait ( decaf::util::concurrent::ConditionHandle \* *condition*, long long *mills*, long long *nanos* ) [static]

Waits for the condition to be signaled or for the time specified to ellapse.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
<i>mills</i>	the time in milliseconds to wait for the condition to be signaled.
<i>nanos</i>	additional time in nanoseconds to wait for the thread to be signaled.

6.214.1.6 static void decaf::internal::util::concurrent::ConditionImpl::wait ( decaf::util::concurrent::ConditionHandle \* *condition* ) [static]

Waits for the condition to be signaled.

#### Parameters

<i>condition</i>	the handle to the condition to wait on.
------------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**ConditionImpl.h**

## 6.215 decaf::net::ConnectException Class Reference

```
#include <src/main/decaf/net/ConnectException.h>
```

Inheritance diagram for decaf::net::ConnectException:

### Public Member Functions

- **ConnectException** () throw ()  
*Default Constructor.*
- **ConnectException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ConnectException** (const **ConnectException** &ex) throw ()  
*Copy Constructor.*
- **ConnectException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ConnectException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ConnectException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ConnectException** \* clone () const  
*Clones this exception.*
- virtual ~**ConnectException** () throw ()

### 6.215.1 Constructor & Destructor Documentation

6.215.1.1 decaf::net::ConnectException::ConnectException ( ) throw () [inline]

Default Constructor.

6.215.1.2 decaf::net::ConnectException::ConnectException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

ex	An exception that should become this type of Exception
----	--



6.215.1.3 `decaf::net::ConnectException::ConnectException ( const ConnectException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.215.1.4 `decaf::net::ConnectException::ConnectException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.215.1.5 `decaf::net::ConnectException::ConnectException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.215.1.6 `decaf::net::ConnectException::ConnectException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.215.1.7 `virtual decaf::net::ConnectException::~~ConnectException ( ) throw ()`  
`[inline, virtual]`

## 6.215.2 Member Function Documentation

6.215.2.1 `virtual ConnectException* decaf::net::ConnectException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3467).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ConnectException.h`

## 6.216 cms::Connection Class Reference

The client's connection to its provider.

```
#include <src/main/cms/Connection.h>
```

Inheritance diagram for cms::Connection:

### Public Member Functions

- virtual **~Connection** ()
- virtual void **close** ()=0 throw ( CMSException )  
*Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).*
- virtual const **ConnectionMetaData** \* **getMetaData** () const =0 throw ( CMSException )  
*Gets the metadata for this connection.*
- virtual **Session** \* **createSession** ()=0 throw ( CMSException )

*Creates an **AUTO\_ACKNOWLEDGE Session** (p. 3305).*

- virtual **Session** \* **createSession** (**Session::AcknowledgeMode** ackMode)=0  
throw ( **CMSEException** )

*Creates a new **Session** (p. 3305) to work for this **Connection** (p. 1232) using the specified acknowledgment mode.*

- virtual std::string **getClientID** () const =0

*Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the **setClientID** method.*

- virtual void **setClientID** (const std::string &clientID)=0

*Sets the client identifier for this connection.*

- virtual **ExceptionListener** \* **getExceptionListener** () const =0

*Gets the registered Exception Listener for this connection.*

- virtual void **setExceptionListener** (**ExceptionListener** \*listener)=0

*Sets the registered Exception Listener for this connection.*

### 6.216.1 Detailed Description

The client's connection to its provider.

Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.
- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 1355) object.
- It supports an optional **ExceptionListener** (p. 1801) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

#### Since

1.0

### 6.216.2 Constructor & Destructor Documentation

6.216.2.1 `virtual cms::Connection::~~Connection ( ) [inline, virtual]`

### 6.216.3 Member Function Documentation

6.216.3.1 `virtual void cms::Connection::close ( ) throw ( CMSEException ) [pure virtual]`

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)
--

Implements **cms::Closeable** (p. 1120).

Implemented in **activemq::core::ActiveMQConnection** (p. 250).

6.216.3.2 `virtual Session* cms::Connection::createSession ( Session::AcknowledgeMode ackMode ) throw ( CMSEException ) [pure virtual]`

Creates a new **Session** (p. 3305) to work for this **Connection** (p. 1232) using the specified acknowledgment mode.

#### Parameters

<i>ackMode</i>	the Acknowledgment Mode to use.
----------------	---------------------------------

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)
--

Implemented in **activemq::core::ActiveMQConnection** (p. 250).

6.216.3.3 virtual **Session\*** cms::Connection::createSession ( ) throw ( **CMSEException** )  
[pure virtual]

Creates an AUTO\_ACKNOWLEDGE **Session** (p. 3305).

#### Exceptions

<b>CMSEException</b> (p. 1130)	
-----------------------------------	--

Implemented in **activemq::core::ActiveMQConnection** (p. 251).

6.216.3.4 virtual std::string cms::Connection::getClientID ( ) const [pure virtual]

Get the Client Id for this session, the client Id is provider specific and is either assigned by the connection factory or set using the setClientID method.

#### Returns

Client Id String for this **Connection** (p. 1232).

#### Exceptions

<b>CMSEException</b> (p. 1130)	if the provider fails to return the client id or an internal error occurs.
-----------------------------------	--

Implemented in **activemq::core::ActiveMQConnection** (p. 252).

6.216.3.5 virtual **ExceptionListener\*** cms::Connection::getExceptionListener ( ) const  
[pure virtual]

Gets the registered Exception Listener for this connection.

#### Returns

pointer to an exception listener or NULL

Implemented in **activemq::core::ActiveMQConnection** (p. 253).

6.216.3.6 virtual const **ConnectionMetaData\*** cms::Connection::getMetaData ( ) const  
throw ( **CMSEException** ) [pure virtual]

Gets the metadata for this connection.

#### Returns

the connection MetaData pointer ( caller does not own it ).

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	if the provider fails to get the connection metadata for this connection.
---	---

**See also****ConnectionMetaData** (p. 1355)**Since**

2.0

Implemented in **activemq::core::ActiveMQConnection** (p. 254).

```
6.216.3.7  virtual void cms::Connection::setClientID ( const std::string & clientID ) [pure
virtual]
```

Sets the client identifier for this connection.

The preferred way to assign a CMS client's client identifier is for it to be configured in a client-specific **ConnectionFactory** (p. 1294) object and transparently assigned to the **Connection** (p. 1232) object it creates.

If a client sets the client identifier explicitly, it must do so immediately after it creates the connection and before any other action on the connection is taken. After this point, setting the client identifier is a programming error that should throw an **IllegalStateException** (p. 1958).

**Parameters**

<i>clientID</i>	The unique client identifier to assign to the <b>Connection</b> (p. 1232).
-----------------	--

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	if the provider fails to set the client id due to some internal error.
<b><i>InvalidClientIDException</i></b>	if the id given is somehow invalid or is a duplicate.
<b><i>IllegalStateException</i></b> (p. 1958)	if the client tries to set the id after a <b>Connection</b> (p. 1232) method has been called.

Implemented in **activemq::core::ActiveMQConnection** (p. 260).

```
6.216.3.8  virtual void cms::Connection::setExceptionListener ( ExceptionListener * listener
) [pure virtual]
```

Sets the registered Exception Listener for this connection.

## Parameters

<i>listener</i>	pointer to and <b>ExceptionListener</b> (p. 1801)
-----------------	---

Implemented in **activemq::core::ActiveMQConnection** (p. 261).

The documentation for this class was generated from the following file:

- src/main/cms/**Connection.h**

## 6.217 activemq::commands::ConnectionControl Class Reference

```
#include <src/main/activemq/commands/ConnectionControl.h>
```

Inheritance diagram for activemq::commands::ConnectionControl:

## Public Member Functions

- **ConnectionControl** ()
- virtual **~ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionControl** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual bool **isExit** () const
- virtual void **setExit** (bool exit)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool faultTolerant)
- virtual bool **isResume** () const
- virtual void **setResume** (bool resume)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool suspend)

- virtual const std::string & **getConnectedBrokers** () const
- virtual std::string & **getConnectedBrokers** ()
- virtual void **setConnectedBrokers** (const std::string &connectedBrokers)
- virtual const std::string & **getReconnectTo** () const
- virtual std::string & **getReconnectTo** ()
- virtual void **setReconnectTo** (const std::string &reconnectTo)
- virtual bool **isRebalanceConnection** () const
- virtual void **setRebalanceConnection** (bool rebalanceConnection)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONCONTROL** = 18

### Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**
- std::string **connectedBrokers**
- std::string **reconnectTo**
- bool **rebalanceConnection**

## 6.217.1 Constructor & Destructor Documentation

6.217.1.1 **activemq::commands::ConnectionControl::ConnectionControl ( )**

6.217.1.2 **virtual activemq::commands::ConnectionControl::~~ConnectionControl ( )**  
[virtual]

## 6.217.2 Member Function Documentation

6.217.2.1 **virtual ConnectionControl\* activemq::commands::ConnectionControl::cloneDataStructure ( )**  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.



Implements **activemq::commands::DataStructure** (p. 1628).

6.217.2.2 `virtual void activemq::commands::ConnectionControl::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.217.2.3 `virtual bool activemq::commands::ConnectionControl::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.217.2.4 `virtual const std::string& activemq::commands::ConnectionControl::getConnectedBrokers ( ) const [virtual]`

6.217.2.5 `virtual std::string& activemq::commands::ConnectionControl::getConnectedBrokers ( ) [virtual]`

6.217.2.6 `virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.217.2.7 `virtual const std::string& activemq::commands::ConnectionControl::getReconnectTo ( ) const [virtual]`

- 6.217.2.8 `virtual std::string& activemq::commands::ConnectionControl::getReconnectTo ( )`  
[virtual]
- 6.217.2.9 `virtual bool activemq::commands::ConnectionControl::isClose ( ) const`  
[virtual]
- 6.217.2.10 `virtual bool activemq::commands::ConnectionControl::isExit ( ) const`  
[virtual]
- 6.217.2.11 `virtual bool activemq::commands::ConnectionControl::isFaultTolerant ( ) const`  
[virtual]
- 6.217.2.12 `virtual bool activemq::commands::ConnectionControl::isRebalanceConnection ( )`  
`const` [virtual]
- 6.217.2.13 `virtual bool activemq::commands::ConnectionControl::isResume ( ) const`  
[virtual]
- 6.217.2.14 `virtual bool activemq::commands::ConnectionControl::isSuspend ( ) const`  
[virtual]
- 6.217.2.15 `virtual void activemq::commands::ConnectionControl::setClose ( bool close )`  
[virtual]
- 6.217.2.16 `virtual void activemq::commands::ConnectionControl::setConnectedBrokers ( const`  
`std::string & connectedBrokers )` [virtual]
- 6.217.2.17 `virtual void activemq::commands::ConnectionControl::setExit ( bool exit )`  
[virtual]
- 6.217.2.18 `virtual void activemq::commands::ConnectionControl::setFaultTolerant ( bool`  
`faultTolerant )` [virtual]
- 6.217.2.19 `virtual void activemq::commands::ConnectionControl::setRebalanceConnection (`  
`bool rebalanceConnection )` [virtual]
- 6.217.2.20 `virtual void activemq::commands::ConnectionControl::setReconnectTo ( const`  
`std::string & reconnectTo )` [virtual]
- 6.217.2.21 `virtual void activemq::commands::ConnectionControl::setResume ( bool resume )`  
[virtual]
- 6.217.2.22 `virtual void activemq::commands::ConnectionControl::setSuspend ( bool suspend )`  
[virtual]

6.217.2.23 `virtual std::string activemq::commands::ConnectionControl::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.217.2.24 `virtual Pointer<Command> activemq::commands::ConnectionControl::visit`  
( `activemq::state::CommandVisitor * visitor` ) throw ( `exceptions::ActiveMQException` ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.217.3 Field Documentation

6.217.3.1 `bool activemq::commands::ConnectionControl::close` [protected]

6.217.3.2 `std::string activemq::commands::ConnectionControl::connectedBrokers`  
[protected]

6.217.3.3 `bool activemq::commands::ConnectionControl::exit` [protected]

6.217.3.4 `bool activemq::commands::ConnectionControl::faultTolerant`  
[protected]

6.217.3.5 `const unsigned char activemq::commands::ConnectionControl::ID_ -`  
`CONNECTIONCONTROL = 18` [static]

6.217.3.6 `bool activemq::commands::ConnectionControl::rebalanceConnection`  
[protected]

6.217.3.7 `std::string activemq::commands::ConnectionControl::reconnectTo`  
[protected]

6.217.3.8 **bool** **activemq::commands::ConnectionControl::resume**  
[protected]

6.217.3.9 **bool** **activemq::commands::ConnectionControl::suspend**  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionControl.h`

## 6.218 **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1242).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

## 6.218

**activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

### Class Reference

1247

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.218.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1242).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.218.2 Constructor & Destructor Documentation

6.218.2.1 **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.218.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.218.3 Member Function Documentation

6.218.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.218.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.218.3.3  virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.218.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.218.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

6.218

**activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

**Class Reference**

1249

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.218.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.218.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionControlMarshaller.h**

## 6.219 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1246).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )



## 6.219

### activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller

#### Class Reference

1251

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

#### 6.219.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1246).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.219.2 Constructor & Destructor Documentation

6.219.2.1 **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.219.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

#### 6.219.3 Member Function Documentation

6.219.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

##### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.219.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

##### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.219.3.3  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.219.3.4  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.219.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

6.219

**activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

**Class Reference**

**1253**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.219.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.219.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionControlMarshaller.h**

## 6.220 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1250).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

## 6.220

### activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller

#### Class Reference

1255

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.220.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1250).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.220.2 Constructor & Destructor Documentation

6.220.2.1 **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.220.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.220.3 Member Function Documentation

6.220.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.220.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.220.3.3  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.220.3.4  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.220.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

6.220

**activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

**Class Reference**

1257

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.220.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.220.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionControlMarshaller.h**

## 6.221 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1254).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )



## 6.221

**activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

### Class Reference

1259

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.221.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1254).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.221.2 Constructor & Destructor Documentation

6.221.2.1 **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.221.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.221.3 Member Function Documentation

6.221.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.221.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.221.3.3  virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.221.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.221.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

6.221

**activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

**Class Reference**

**1261**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.221.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.221.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConnectionControlMarshaller.h**

## 6.222 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1258).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

## 6.222

**activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller**

**Class Reference**

**1263**

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.222.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1258).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.222.2 Constructor & Destructor Documentation

6.222.2.1 **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.222.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.222.3 Member Function Documentation

6.222.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.222.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.222.3.3  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

```
6.222.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.222.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

## 6.222

**activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller****Class Reference**

1265

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.222.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.222.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionControlMarshaller.h**

## 6.223 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1262).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**:

### Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )



## 6.223

**activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

**Class Reference**

**1267**

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.223.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1262).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.223.2 Constructor & Destructor Documentation

6.223.2.1 **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::ConnectionControlMarshaller**  
( ) [inline]

6.223.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::~~ConnectionControlMarshaller**  
( ) [inline, virtual]

### 6.223.3 Member Function Documentation

6.223.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.223.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```

6.223.3.3  virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```

6.223.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```

6.223.3.5  virtual int activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

## 6.223

**activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller****Class Reference**

1269

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.223.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.223.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionControlMarshaller.h**

## 6.224 activemq::commands::ConnectionError Class Reference

```
#include <src/main/activemq/commands/ConnectionError.h>
```

Inheritance diagram for **activemq::commands::ConnectionError**:

### Public Member Functions

- **ConnectionError** ()
- virtual **~ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaller share.*
- virtual **ConnectionError** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONERROR** = 16

### Protected Attributes

- **Pointer**< **BrokerError** > **exception**
- **Pointer**< **ConnectionId** > **connectionId**

## 6.224.1 Constructor & Destructor Documentation

6.224.1.1 **activemq::commands::ConnectionError::ConnectionError** ( )

6.224.1.2 **virtual activemq::commands::ConnectionError::~~ConnectionError** ( )  
[virtual]

## 6.224.2 Member Function Documentation

6.224.2.1 **virtual ConnectionError\* activemq::commands::ConnectionError::cloneDataStructure**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.224.2.2 **virtual void activemq::commands::ConnectionError::copyDataStructure** ( const  
**DataStructure** \* *src* ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.224.2.3 **virtual bool** **activemq::commands::ConnectionError::equals** ( **const DataStructure**  
\* *value* ) **const** [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.224.2.4 **virtual const Pointer<ConnectionId>&** **activemq::commands::ConnectionError::getConnectionId** ( ) **const**  
[virtual]

6.224.2.5 **virtual Pointer<ConnectionId>&** **activemq::commands::ConnectionError::getConnectionId** ( )  
[virtual]

6.224.2.6 **virtual unsigned char** **activemq::commands::ConnectionError::getDataStructureType** ( ) **const** [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.224.2.7 **virtual Pointer<BrokerError>&** **activemq::commands::ConnectionError::getException** ( )  
[virtual]

6.224.2.8 **virtual const Pointer<BrokerError>&** **activemq::commands::ConnectionError::getException** ( ) **const**  
[virtual]

6.224.2.9 **virtual void** **activemq::commands::ConnectionError::setConnectionId** ( **const Pointer< ConnectionId > & connectionId** ) [virtual]

6.224.2.10 `virtual void activemq::commands::ConnectionError::setException ( const Pointer< BrokerError > & exception ) [virtual]`

6.224.2.11 `virtual std::string activemq::commands::ConnectionError::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.224.2.12 `virtual Pointer<Command> activemq::commands::ConnectionError::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

## 6.224.3 Field Documentation

6.224.3.1 `Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId [protected]`

6.224.3.2 `Pointer<BrokerError> activemq::commands::ConnectionError::exception [protected]`

6.224.3.3 `const unsigned char activemq::commands::ConnectionError::ID_ - CONNECTIONERROR = 16 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionError.h**

## 6.225 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1270).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.225.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1270).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.225.2 Constructor & Destructor Documentation

6.225.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::ConnectionErrorMarshaller ( ) [inline]`

6.225.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller ( ) [inline, virtual]`

## 6.225.3 Member Function Documentation

6.225.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.225.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.225.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.225.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.225.3.5  virtual int activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.225 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller

### Class Reference 1277

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.225.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.225.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionErrorMarshaller.h**

## 6.226 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1274).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.226.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1274).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.226.2 Constructor & Destructor Documentation

6.226.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::ConnectionErrorMarshaller ( ) [inline]`

6.226.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller ( ) [inline, virtual]`

## 6.226.3 Member Function Documentation

6.226.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.226.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.226.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.226.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.226.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.226 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller

### Class Reference 1281

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.226.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.226.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionErrorMarshaller.h**

## 6.227 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1278).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.227.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1278).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.227.2 Constructor & Destructor Documentation

6.227.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::ConnectionErrorMarshaller`  
`( ) [inline]`

6.227.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller`  
`( ) [inline, virtual]`

## 6.227.3 Member Function Documentation

6.227.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::createObject (`  
`) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.227.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::getDataStructureType`  
`( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.227.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseMarshal`  
`( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw (`  
`decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.227.3.4  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.227.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.227 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller

### Class Reference 1285

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.227.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.227.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionErrorMarshaller.h**

## 6.228 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1282).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.228.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1282).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.228.2 Constructor & Destructor Documentation

6.228.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::ConnectionErrorMarshaller ( ) [inline]`

6.228.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller ( ) [inline, virtual]`

## 6.228.3 Member Function Documentation

6.228.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.228.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.228.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.228.3.4  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.228.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.228 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller

### Class Reference 1289

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.228.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.228.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionErrorMarshaller.h**

## 6.229 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1286).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.229.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1286).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.229.2 Constructor & Destructor Documentation

6.229.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::ConnectionErrorMarshaller ( ) [inline]`

6.229.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller ( ) [inline, virtual]`

## 6.229.3 Member Function Documentation

6.229.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.229.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.229.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.229.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.229.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.229 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller

### Class Reference 1293

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.229.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.229.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConnectionErrorMarshaller.h**

## 6.230 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1290).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller:

### Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.230.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1290).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.230.2 Constructor & Destructor Documentation

6.230.2.1 `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::ConnectionErrorMarshaller ( ) [inline]`

6.230.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller ( ) [inline, virtual]`

## 6.230.3 Member Function Documentation

6.230.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.230.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.230.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

```
6.230.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.230.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.230 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller

### Class Reference 1297

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.230.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.230.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionErrorMarshaller.h**

## 6.231 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p. 1232) objects returned implement the CMS **Connection** (p. 1232) interface and hide the CMS Provider specific implementation details behind that interface.

```
#include <src/main/cms/ConnectionFactory.h>
```

Inheritance diagram for cms::ConnectionFactory:

### Public Member Functions

- virtual **~ConnectionFactory** ()
- virtual **Connection** \* **createConnection** ()=0 throw ( CMSEException )  
*Creates a connection with the default user identity.*
- virtual **cms::Connection** \* **createConnection** (const std::string &username, const std::string &password)=0 throw ( cms::CMSEException )  
*Creates a connection with the default specified identity.*
- virtual **cms::Connection** \* **createConnection** (const std::string &username, const std::string &password, const std::string &clientId)=0 throw ( cms::CMSEException )  
*Creates a connection with the specified user identity.*

### Static Public Member Functions

- static **ConnectionFactory** \* **createCMSConnectionFactory** (const std::string &brokerURI) throw ( cms::CMSEException )  
*Static method that is used to create a provider specific connection factory.*

#### 6.231.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p. 1232) objects returned implement the CMS **Connection** (p. 1232) interface and hide the CMS Provider specific implementation details behind that interface.

A Client creates a new **ConnectionFactory** (p. 1294) either directly by instantiating the provider specific implementation of the factory or by using the static method `createCMSConnectionFactory` which all providers are required to implement.

#### Since

1.0



## 6.231.2 Constructor & Destructor Documentation

6.231.2.1 `virtual cms::ConnectionFactory::~~ConnectionFactory ( ) [inline, virtual]`

## 6.231.3 Member Function Documentation

6.231.3.1 `static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory ( const std::string & brokerURI ) throw ( cms::CMSEException ) [static]`

Static method that is used to create a provider specific connection factory.

The provider implements this method in their library and returns an instance of a **ConnectionFactory** (p. 1294) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

### Parameters

<i>brokerURI</i>	The remote address to use to connect to the Provider.
------------------	---

### Returns

A pointer to a provider specific implementation of the **ConnectionFactory** (p. 1294) interface, the caller is responsible for deleting this resource.

### Exceptions

<b>CMSEException</b> (p. 1130)	if an internal error occurs while creating the <b>ConnectionFactory</b> (p. 1294).
-----------------------------------	--

6.231.3.2 `virtual cms::Connection* cms::ConnectionFactory::createConnection ( const std::string & username, const std::string & password ) throw ( cms::CMSEException ) [pure virtual]`

Creates a connection with the default specified identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3527) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

### Parameters

<i>username</i>	The user name used to authenticate with the Provider.
<i>password</i>	The password used to authenticate with the Provider.

### Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

**Exceptions**

<b>CMSEException</b> (p. 1130)	if an internal error occurs while creating the <b>Connection</b> (p. 1232).
-----------------------------------	---

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 267).

```
6.231.3.3  virtual cms::Connection* cms::ConnectionFactory::createConnection ( const
std::string & username, const std::string & password, const std::string & clientId )
throw ( cms::CMSEException )  [pure virtual]
```

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3527) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

**Parameters**

<i>username</i>	The user name used to authenticate with the Provider.
<i>password</i>	The password used to authenticate with the Provider.
<i>clientId</i>	The Client Id assigned to connection. If the id is the empty string ("") then a random client Id is created for this connection.

**Returns**

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

**Exceptions**

<b>CMSEException</b> (p. 1130)	if an internal error occurs while creating the <b>Connection</b> (p. 1232).
-----------------------------------	---

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 268).

```
6.231.3.4  virtual Connection* cms::ConnectionFactory::createConnection ( ) throw (
CMSEException )  [pure virtual]
```

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 3527) method is explicitly called.

**Returns**

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

**Exceptions**

<b>CMSException</b> (p. 1130)	if an internal error occurs while creating the <b>Connection</b> (p. 1232).
----------------------------------	---

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 267).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionFactory.h**

**6.232 activemq::commands::ConnectionId Class Reference**

```
#include <src/main/activemq/commands/ConnectionId.h>
```

Inheritance diagram for **activemq::commands::ConnectionId**:

**Public Types**

- typedef **decaf::lang::PointerComparator**< **ConnectionId** > **COMPARATOR**

**Public Member Functions**

- **ConnectionId** ()
- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** \*sessionId)
- **ConnectionId** (const **ProducerId** \*producerId)
- **ConnectionId** (const **ConsumerId** \*consumerId)
- virtual ~**ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const

- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &**value**)
- virtual int **compareTo** (const **ConnectionId** &**value**) const
- virtual bool **equals** (const **ConnectionId** &**value**) const
- virtual bool **operator==** (const **ConnectionId** &**value**) const
- virtual bool **operator<** (const **ConnectionId** &**value**) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONID** = 120

### Protected Attributes

- std::string **value**

## 6.232.1 Member Typedef Documentation

- 6.232.1.1 `typedef decaf::lang::PointerComparator<ConnectionId>  
activemq::commands::ConnectionId::COMPARATOR`

## 6.232.2 Constructor & Destructor Documentation

- 6.232.2.1 `activemq::commands::ConnectionId::ConnectionId ( )`
- 6.232.2.2 `activemq::commands::ConnectionId::ConnectionId ( const ConnectionId & other )`
- 6.232.2.3 `activemq::commands::ConnectionId::ConnectionId ( const SessionId * sessionId )`
- 6.232.2.4 `activemq::commands::ConnectionId::ConnectionId ( const ProducerId * producerId )`
- 6.232.2.5 `activemq::commands::ConnectionId::ConnectionId ( const ConsumerId * consumerId )`
- 6.232.2.6 `virtual activemq::commands::ConnectionId::~~ConnectionId ( ) [virtual]`

## 6.232.3 Member Function Documentation

- 6.232.3.1 `virtual ConnectionId* activemq::commands::ConnectionId::cloneDataStructure ( )  
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.232.3.2 `virtual int activemq::commands::ConnectionId::compareTo ( const ConnectionId & value ) const` [virtual]

6.232.3.3 `virtual void activemq::commands::ConnectionId::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<code>src</code>	- Source Object
------------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.232.3.4 `virtual bool activemq::commands::ConnectionId::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.232.3.5 `virtual bool activemq::commands::ConnectionId::equals ( const ConnectionId & value ) const` [virtual]

6.232.3.6 `virtual unsigned char activemq::commands::ConnectionId::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

- 6.232.3.7 `virtual const std::string& activemq::commands::ConnectionId::getValue ( ) const`  
[virtual]
- 6.232.3.8 `virtual std::string& activemq::commands::ConnectionId::getValue ( )`  
[virtual]
- 6.232.3.9 `virtual bool activemq::commands::ConnectionId::operator< ( const ConnectionId  
& value ) const` [virtual]
- 6.232.3.10 `ConnectionId& activemq::commands::ConnectionId::operator= ( const  
ConnectionId & other )`
- 6.232.3.11 `virtual bool activemq::commands::ConnectionId::operator== ( const ConnectionId  
& value ) const` [virtual]
- 6.232.3.12 `virtual void activemq::commands::ConnectionId::setValue ( const std::string & value  
)` [virtual]
- 6.232.3.13 `virtual std::string activemq::commands::ConnectionId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 796).

### 6.232.4 Field Documentation

- 6.232.4.1 `const unsigned char activemq::commands::ConnectionId::ID_  
CONNECTIONID = 120` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.232.4.2 `std::string activemq::commands::ConnectionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionId.h`

## 6.233 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1301).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.233.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1301).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.233.2 Constructor & Destructor Documentation

6.233.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]`

6.233.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::~~ConnectionIdMarshaller ( ) [inline, virtual]`

## 6.233.3 Member Function Documentation

6.233.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.233.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.233.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.233.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseUnmarshal  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.233.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal1  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```

6.233.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```

6.233.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionIdMarshaller.h**

## 6.234 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1305).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.234.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1305).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.234.2 Constructor & Destructor Documentation

6.234.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]`

6.234.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::~~ConnectionIdMarshaller ( ) [inline, virtual]`

## 6.234.3 Member Function Documentation

6.234.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.234.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.234.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.234.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseUnmarshal  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.234.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal1  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.234.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.234.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionIdMarshaller.h**

## 6.235 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1309).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.235.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1309).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.235.2 Constructor & Destructor Documentation

6.235.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]`

6.235.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::~~ConnectionIdMarshaller ( ) [inline, virtual]`

## 6.235.3 Member Function Documentation

6.235.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.235.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.235.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.235.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.235.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.235.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.235.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionIdMarshaller.h**

## 6.236 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1313).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.236.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1313).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.236.2 Constructor & Destructor Documentation

6.236.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]`

6.236.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::~~ConnectionIdMarshaller ( ) [inline, virtual]`

## 6.236.3 Member Function Documentation

6.236.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.236.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.236.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.236.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.236.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```

6.236.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```

6.236.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionIdMarshaller.h**

## 6.237 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1317).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.237.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1317).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.237.2 Constructor & Destructor Documentation

6.237.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]`

6.237.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::~~ConnectionIdMarshaller ( ) [inline, virtual]`

## 6.237.3 Member Function Documentation

6.237.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.237.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.237.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.237.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseUnmarshal  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.237.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal1  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```

6.237.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```

6.237.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h

## 6.238 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1321).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller:

### Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.238.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1321).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.238.2 Constructor & Destructor Documentation

6.238.2.1 `activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::ConnectionIdMarshaller ( ) [inline]`

6.238.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::~~ConnectionIdMarshaller ( ) [inline, virtual]`

## 6.238.3 Member Function Documentation

6.238.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.238.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.238.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.238.3.4 **virtual void** **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
[virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.238.3.5 **virtual int** **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )**  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.238.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.238.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionIdMarshaller.h**

## 6.239 activemq::commands::ConnectionInfo Class Reference

```
#include <src/main/activemq/commands/ConnectionInfo.h>
```

Inheritance diagram for activemq::commands::ConnectionInfo:

## Public Member Functions

- **ConnectionInfo** ()
- virtual **~ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConnectionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &password)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &userName)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool brokerMasterConnector)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool manageable)
- virtual bool **isClientMaster** () const
- virtual void **setClientMaster** (bool clientMaster)

- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool **faultTolerant**)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONNECTIONINFO** = 3

### Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector< decaf::lang::Pointer< **BrokerId** > > **brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**
- bool **faultTolerant**

## 6.239.1 Constructor & Destructor Documentation

6.239.1.1 **activemq::commands::ConnectionInfo::ConnectionInfo** ( )

6.239.1.2 **virtual activemq::commands::ConnectionInfo::~~ConnectionInfo** ( ) [virtual]

## 6.239.2 Member Function Documentation

6.239.2.1 **virtual ConnectionInfo\*** **activemq::commands::ConnectionInfo::cloneDataStructure**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).



6.239.2.2 `virtual void activemq::commands::ConnectionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 724).

6.239.2.3 `Pointer<RemoveInfo> activemq::commands::ConnectionInfo::createRemoveCommand ( ) const`

6.239.2.4 `virtual bool activemq::commands::ConnectionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 725).

6.239.2.5 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath ( ) const [virtual]`

6.239.2.6 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath ( ) [virtual]`

6.239.2.7 `virtual const std::string& activemq::commands::ConnectionInfo::getClientId ( ) const [virtual]`

6.239.2.8 `virtual std::string& activemq::commands::ConnectionInfo::getClientId ( ) [virtual]`

6.239.2.9 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId ( ) [virtual]`

6.239.2.10 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId ( ) const [virtual]`

6.239.2.11 `virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.239.2.12 `virtual const std::string& activemq::commands::ConnectionInfo::getPassword ( ) const [virtual]`

6.239.2.13 `virtual std::string& activemq::commands::ConnectionInfo::getPassword ( ) [virtual]`

6.239.2.14 `virtual const std::string& activemq::commands::ConnectionInfo::getUserName ( ) const [virtual]`

6.239.2.15 `virtual std::string& activemq::commands::ConnectionInfo::getUserName ( ) [virtual]`

6.239.2.16 `virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector ( ) const [virtual]`

6.239.2.17 `virtual bool activemq::commands::ConnectionInfo::isClientMaster ( ) const [virtual]`

6.239.2.18 `virtual bool activemq::commands::ConnectionInfo::isConnectionInfo ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isConnectionInfo()** (p. 1328) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 726).

6.239.2.19 `virtual bool activemq::commands::ConnectionInfo::isFaultTolerant ( ) const [virtual]`

6.239.2.20 `virtual bool activemq::commands::ConnectionInfo::isManageable ( ) const [virtual]`

6.239.2.21 `virtual void activemq::commands::ConnectionInfo::setBrokerMasterConnector ( bool brokerMasterConnector ) [virtual]`

- 6.239.2.22 `virtual void activemq::commands::ConnectionInfo::setBrokerPath ( const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath ) [virtual]`
- 6.239.2.23 `virtual void activemq::commands::ConnectionInfo::setClientId ( const std::string & clientId ) [virtual]`
- 6.239.2.24 `virtual void activemq::commands::ConnectionInfo::setClientMaster ( bool clientMaster ) [virtual]`
- 6.239.2.25 `virtual void activemq::commands::ConnectionInfo::setConnectionId ( const Pointer< ConnectionId > & connectionId ) [virtual]`
- 6.239.2.26 `virtual void activemq::commands::ConnectionInfo::setFaultTolerant ( bool faultTolerant ) [virtual]`
- 6.239.2.27 `virtual void activemq::commands::ConnectionInfo::setManageable ( bool manageable ) [virtual]`
- 6.239.2.28 `virtual void activemq::commands::ConnectionInfo::setPassword ( const std::string & password ) [virtual]`
- 6.239.2.29 `virtual void activemq::commands::ConnectionInfo::setUserName ( const std::string & userName ) [virtual]`
- 6.239.2.30 `virtual std::string activemq::commands::ConnectionInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

- 6.239.2.31 `virtual Pointer<Command> activemq::commands::ConnectionInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.239.3 Field Documentation

- 6.239.3.1 `bool activemq::commands::ConnectionInfo::brokerMasterConnector`  
[protected]
- 6.239.3.2 `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::ConnectionInfo::brokerPath` [protected]
- 6.239.3.3 `std::string activemq::commands::ConnectionInfo::clientId`  
[protected]
- 6.239.3.4 `bool activemq::commands::ConnectionInfo::clientMaster`  
[protected]
- 6.239.3.5 `Pointer<ConnectionId> activemq::commands::ConnectionInfo::connectionId`  
[protected]
- 6.239.3.6 `bool activemq::commands::ConnectionInfo::faultTolerant`  
[protected]
- 6.239.3.7 `const unsigned char activemq::commands::ConnectionInfo::ID_-`  
`CONNECTIONINFO = 3` [static]
- 6.239.3.8 `bool activemq::commands::ConnectionInfo::manageable`  
[protected]
- 6.239.3.9 `std::string activemq::commands::ConnectionInfo::password`  
[protected]
- 6.239.3.10 `std::string activemq::commands::ConnectionInfo::userName`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionInfo.h`

## 6.240 `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1330).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller`:

## Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.240.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1330).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.240.2 Constructor & Destructor Documentation

6.240.2.1 **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::ConnectionInfoMarshaller**  
( ) [inline]

6.240.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller**  
( ) [inline, virtual]

### 6.240.3 Member Function Documentation

6.240.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.240.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.240.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.240.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
 (p. 766).

6.240.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal1  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
 (p. 767).

6.240.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 768).

6.240.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 769).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h`



## 6.241 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1335).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.241.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1335).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.241.2 Constructor & Destructor Documentation

6.241.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::ConnectionInfoMarshaller ( ) [inline]`

6.241.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller ( ) [inline, virtual]`

## 6.241.3 Member Function Documentation

6.241.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.241.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.241.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.241.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.241.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.241.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.241.3.7  virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionInfoMarshaller.h**

## 6.242 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1339).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.242.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1339).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.242.2 Constructor & Destructor Documentation

6.242.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::ConnectionInfoMarshaller ( ) [inline]`

6.242.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller ( ) [inline, virtual]`

## 6.242.3 Member Function Documentation

6.242.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.242.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.242.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.242.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

6.242.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.242.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.242.3.7  virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConnectionInfoMarshaller.h**



## 6.243 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1343).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.243.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1343).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.243.2 Constructor & Destructor Documentation

6.243.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::ConnectionInfoMarshaller ( ) [inline]`

6.243.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller ( ) [inline, virtual]`

## 6.243.3 Member Function Documentation

6.243.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.243.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.243.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.243.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

6.243.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.243.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.243.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionInfoMarshaller.h**

## 6.244 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1347).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.244.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1347).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.244.2 Constructor & Destructor Documentation

6.244.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::ConnectionInfoMarshaller ( ) [inline]`

6.244.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller ( ) [inline, virtual]`

## 6.244.3 Member Function Documentation

6.244.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.244.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.244.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.244.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

6.244.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.244.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.244.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConnectionInfoMarshaller.h**



## 6.245 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1351).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller:

### Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.245.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1351).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.245.2 Constructor & Destructor Documentation

6.245.2.1 `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::ConnectionInfoMarshaller ( ) [inline]`

6.245.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::~~ConnectionInfoMarshaller ( ) [inline, virtual]`

## 6.245.3 Member Function Documentation

6.245.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.245.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.245.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.245.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.245.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.245.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.245.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConnectionInfoMarshaller.h**

## 6.246 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p. 1355) object provides information describing the **Connection** (p. 1232) object.

```
#include <src/main/cms/ConnectionMetaData.h>
```

Inheritance diagram for cms::ConnectionMetaData:

### Public Member Functions

- virtual **~ConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS API version.*
- virtual int **getCMSMajorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS major version number.*
- virtual int **getCMSMinorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS minor version number.*
- virtual std::string **getCMSProviderName** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider name.*
- virtual std::string **getProviderVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider version.*
- virtual int **getProviderMajorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider major version number.*
- virtual int **getProviderMinorVersion** () const =0 throw ( cms::CMSEException )  
*Gets the CMS provider minor version number.*
- virtual std::vector< std::string > **getCMSXPropertyNames** () const =0 throw ( cms::CMSEException )  
*Gets an Vector of the CMSX property names.*

### 6.246.1 Detailed Description

A **ConnectionMetaData** (p. 1355) object provides information describing the **Connection** (p. 1232) object.

Since

1.3

## 6.246.2 Constructor & Destructor Documentation

6.246.2.1 `virtual cms::ConnectionMetaData::~~ConnectionMetaData ( ) [inline, virtual]`

## 6.246.3 Member Function Documentation

6.246.3.1 `virtual int cms::ConnectionMetaData::getCMSMajorVersion ( ) const throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS major version number.

### Returns

the CMS API major version number

### Exceptions

<b>CMSEException</b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
-----------------------------------	--

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 276).

6.246.3.2 `virtual int cms::ConnectionMetaData::getCMSMinorVersion ( ) const throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS minor version number.

### Returns

the CMS API minor version number

### Exceptions

<b>CMSEException</b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
-----------------------------------	--

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 276).

6.246.3.3 `virtual std::string cms::ConnectionMetaData::getCMSProviderName ( ) const throw ( cms::CMSEException ) [pure virtual]`

Gets the CMS provider name.

### Returns

the CMS provider name

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 277).

6.246.3.4 `virtual std::string cms::ConnectionMetaData::getCMSVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS API version.

**Returns**

the CMS API Version in String form.

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 277).

6.246.3.5 `virtual std::vector<std::string> cms::ConnectionMetaData::getCMSXPropertyNames ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets an Vector of the CMSX property names.

**Returns**

an Vector of CMSX property names

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 277).

6.246.3.6 `virtual int cms::ConnectionMetaData::getProviderMajorVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider major version number.

**Returns**

the CMS provider major version number

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 278).

6.246.3.7 `virtual int cms::ConnectionMetaData::getProviderMinorVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider minor version number.

**Returns**

the CMS provider minor version number

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 278).

6.246.3.8 `virtual std::string cms::ConnectionMetaData::getProviderVersion ( ) const throw ( cms::CMSException ) [pure virtual]`

Gets the CMS provider version.

**Returns**

the CMS provider version

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	If the CMS Provider fails to retrieve the metadata due to some internal error.
---	--

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 278).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionMetaData.h`

**6.247 activemq::state::ConnectionState Class Reference**

```
#include <src/main/activemq/state/ConnectionState.h>
```



## Public Member Functions

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)
- virtual **~ConnectionState** ()
- std::string **toString** () const
- const **Pointer**< **commands::ConnectionInfo** > & **getInfo** () const
- void **checkShutdown** () const
- void **shutdown** ()
- void **reset** (const **Pointer**< **ConnectionInfo** > &info)
- void **addTempDestination** (const **Pointer**< **DestinationInfo** > &info)
- void **removeTempDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- void **addTransactionState** (const **Pointer**< **TransactionId** > &id)
- const **Pointer**< **TransactionState** > & **getTransactionState** (const **Pointer**< **TransactionId** > &id) const
- std::vector< **Pointer**< **TransactionState** > > **getTransactionStates** () const
- **Pointer**< **TransactionState** > **removeTransactionState** (const **Pointer**< **TransactionId** > &id)
- void **addSession** (const **Pointer**< **SessionInfo** > &info)
- **Pointer**< **SessionState** > **removeSession** (const **Pointer**< **SessionId** > &id)
- const **Pointer**< **SessionState** > & **getSessionState** (const **Pointer**< **SessionId** > &id) const
- const **StlList**< **Pointer**< **DestinationInfo** > > & **getTempDesinations** () const
- std::vector< **Pointer**< **SessionState** > > **getSessionStates** () const
- **StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > **getRecoveringPullConsumers** ()
- void **setConnectionInterruptProcessingComplete** (bool connectionInterruptProcessingComplete)
- bool **isConnectionInterruptProcessingComplete** ()

### 6.247.1 Constructor & Destructor Documentation

- 6.247.1.1 **activemq::state::ConnectionState::ConnectionState** ( const **Pointer**< **ConnectionInfo** > & *info* )
- 6.247.1.2 virtual **activemq::state::ConnectionState::~~ConnectionState** ( ) [virtual]

### 6.247.2 Member Function Documentation

- 6.247.2.1 void **activemq::state::ConnectionState::addSession** ( const **Pointer**< **SessionInfo** > & *info* ) [inline]
- 6.247.2.2 void **activemq::state::ConnectionState::addTempDestination** ( const **Pointer**< **DestinationInfo** > & *info* ) [inline]
- 6.247.2.3 void **activemq::state::ConnectionState::addTransactionState** ( const **Pointer**< **TransactionId** > & *id* ) [inline]

6.247.2.4 `void activemq::state::ConnectionState::checkShutdown ( ) const`

6.247.2.5 `const Pointer<commands::ConnectionInfo>&  
activemq::state::ConnectionState::getInfo ( ) const [inline]`

6.247.2.6 `StlMap< Pointer<ConsumerId>, Pointer<ConsumerInfo>,  
ConsumerId::COMPARATOR > ac-  
tivemq::state::ConnectionState::getRecoveringPullConsumers ( ) [inline]`

6.247.2.7 `const Pointer<SessionState>& ac-  
tivemq::state::ConnectionState::getSessionState ( const  
Pointer< SessionId > & id ) const [inline]`

6.247.2.8 `std::vector< Pointer<SessionState> >  
activemq::state::ConnectionState::getSessionStates ( ) const  
[inline]`

6.247.2.9 `const StlList< Pointer<DestinationInfo> >&  
activemq::state::ConnectionState::getTempDesinations ( ) const [inline]`

6.247.2.10 `const Pointer<TransactionState>& ac-  
tivemq::state::ConnectionState::getTransactionState ( const Pointer<  
TransactionId > & id ) const [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

6.247.2.11 `std::vector< Pointer<TransactionState> >  
activemq::state::ConnectionState::getTransactionStates ( ) const [inline]`

6.247.2.12 `bool activemq::state::ConnectionState::isConnectionInterruptProcessingComplete ( ) [inline]`

6.247.2.13 `Pointer<SessionState> activemq::state::ConnectionState::removeSession ( const Pointer< SessionId > & id ) [inline]`

6.247.2.14 `void activemq::state::ConnectionState::removeTempDestination ( const Pointer<  
ActiveMQDestination > & destination ) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

6.247.2.15 `Pointer<TransactionState> ac-  
tivemq::state::ConnectionState::removeTransactionState ( const Pointer< TransactionId > & id ) [inline]`

6.247.2.16 `void activemq::state::ConnectionState::reset ( const Pointer< ConnectionInfo > & info )`

6.247.2.17 void activemq::state::ConnectionState::setConnectionInterruptProcessingComplete ( bool *connectionInterruptProcessingComplete* ) [inline]

6.247.2.18 void activemq::state::ConnectionState::shutdown ( )

6.247.2.19 std::string activemq::state::ConnectionState::toString ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/ConnectionState.h

## 6.248 activemq::state::ConnectionStateTracker Class Reference

```
#include <src/main/activemq/state/ConnectionStateTracker.h>
```

Inheritance diagram for activemq::state::ConnectionStateTracker:

### Public Member Functions

- **ConnectionStateTracker** ( )
- virtual **~ConnectionStateTracker** ( )
- **Pointer< Tracked > track** (const **Pointer< Command >** &command) throw ( decaf::io::IOException )
- void **trackBack** (const **Pointer< Command >** &command)
- void **restore** (const **Pointer< transport::Transport >** &transport) throw ( decaf::io::IOException )
- void **connectionInterruptProcessingComplete** (transport::Transport \*transport, const **Pointer< ConnectionId >** &connectionId)
- void **transportInterrupted** ( )
- virtual **Pointer< Command > processDestinationInfo** (DestinationInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveDestination** (DestinationInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processProducerInfo** (ProducerInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveProducer** (ProducerId \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processConsumerInfo** (ConsumerInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveConsumer** (ConsumerId \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processSessionInfo** (SessionInfo \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveSession** (SessionId \*id) throw ( exceptions::ActiveMQException )

- virtual **Pointer< Command > processConnectionInfo** (**ConnectionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRemoveConnection** (**ConnectionId** \*id) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processMessage** (**Message** \*message) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processMessageAck** (**MessageAck** \*ack) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processBeginTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processPrepareTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processCommitTransactionOnePhase** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processCommitTransactionTwoPhase** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processRollbackTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- virtual **Pointer< Command > processEndTransaction** (**TransactionInfo** \*info) throw ( exceptions::ActiveMQException )
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)
- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const
- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool trackTransactionProducers)

## Friends

- class **RemoveTransactionAction**

## 6.248.1 Constructor & Destructor Documentation

### 6.248.1.1 activemq::state::ConnectionStateTracker::ConnectionStateTracker ( )

6.248.1.2 `virtual activemq::state::ConnectionStateTracker::~~ConnectionStateTracker ( )`  
`[virtual]`

## 6.248.2 Member Function Documentation

6.248.2.1 `void activemq::state::ConnectionStateTracker::connectionInterruptProcessingComplete ( transport::Transport * transport, const Pointer< ConnectionId > & connectionId )`

6.248.2.2 `int activemq::state::ConnectionStateTracker::getMaxCacheSize ( ) const`  
`[inline]`

6.248.2.3 `bool activemq::state::ConnectionStateTracker::isRestoreConsumers ( ) const`  
`[inline]`

6.248.2.4 `bool activemq::state::ConnectionStateTracker::isRestoreProducers ( ) const`  
`[inline]`

6.248.2.5 `bool activemq::state::ConnectionStateTracker::isRestoreSessions ( ) const`  
`[inline]`

6.248.2.6 `bool activemq::state::ConnectionStateTracker::isRestoreTransaction ( ) const`  
`[inline]`

6.248.2.7 `bool activemq::state::ConnectionStateTracker::isTrackMessages ( ) const`  
`[inline]`

6.248.2.8 `bool activemq::state::ConnectionStateTracker::isTrackTransactionProducers ( ) const`  
`[inline]`

6.248.2.9 `bool activemq::state::ConnectionStateTracker::isTrackTransactions ( ) const`  
`[inline]`

6.248.2.10 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processBeginTransaction ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
`[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1173).

6.248.2.11 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionOnePhase ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
`[virtual]`

Implements `activemq::state::CommandVisitor` (p. 1174).

6.248.2.12 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase ( TransactionInfo \* *info* ) throw ( exceptions::ActiveMQException )**  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1174).

6.248.2.13 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo ( ConnectionInfo \* *info* ) throw ( exceptions::ActiveMQException )**  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1174).

6.248.2.14 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo ( ConsumerInfo \* *info* ) throw ( exceptions::ActiveMQException )**  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1175).

6.248.2.15 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo ( DestinationInfo \* *info* ) throw ( exceptions::ActiveMQException )**  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1175).

6.248.2.16 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction ( TransactionInfo \* *info* ) throw ( exceptions::ActiveMQException )**  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1175).

6.248.2.17 **virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage ( Message \* *message* ) throw ( exceptions::ActiveMQException )**  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1175).

6.248.2.18 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck ( MessageAck \* *ack* ) throw ( exceptions::ActiveMQException )  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1176).

6.248.2.19 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepareTransaction ( TransactionInfo \* *info* ) throw ( exceptions::ActiveMQException )  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1176).

6.248.2.20 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo ( ProducerInfo \* *info* ) throw ( exceptions::ActiveMQException )  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1176).

6.248.2.21 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConnection ( ConnectionId \* *id* ) throw ( exceptions::ActiveMQException )  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1177).

6.248.2.22 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer ( ConsumerId \* *id* ) throw ( exceptions::ActiveMQException ) [virtual]

Implements **activemq::state::CommandVisitor** (p. 1177).

6.248.2.23 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination ( DestinationInfo \* *info* ) throw ( exceptions::ActiveMQException )  
[virtual]

Implements **activemq::state::CommandVisitor** (p. 1177).

6.248.2.24 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer ( ProducerId * id ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1177).

6.248.2.25 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession ( SessionId * id ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1177).

6.248.2.26 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction ( TransactionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1178).

6.248.2.27 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo ( SessionInfo * info ) throw ( exceptions::ActiveMQException )`  
[virtual]

Implements `activemq::state::CommandVisitor` (p. 1178).

6.248.2.28 `void activemq::state::ConnectionStateTracker::restore ( const Pointer<transport::Transport> & transport ) throw ( decaf::io::IOException )`

6.248.2.29 `void activemq::state::ConnectionStateTracker::setMaxCacheSize ( int maxCacheSize )` [inline]

6.248.2.30 `void activemq::state::ConnectionStateTracker::setRestoreConsumers ( bool restoreConsumers )` [inline]

6.248.2.31 `void activemq::state::ConnectionStateTracker::setRestoreProducers ( bool restoreProducers )` [inline]

6.248.2.32 `void activemq::state::ConnectionStateTracker::setRestoreSessions ( bool restoreSessions )` [inline]

6.248.2.33 `void activemq::state::ConnectionStateTracker::setRestoreTransaction ( bool restoreTransaction )` [inline]



- 6.248.2.34 void activemq::state::ConnectionStateTracker::setTrackMessages ( bool *trackMessages* ) [inline]
- 6.248.2.35 void activemq::state::ConnectionStateTracker::setTrackTransactionProducers ( bool *trackTransactionProducers* ) [inline]
- 6.248.2.36 void activemq::state::ConnectionStateTracker::setTrackTransactions ( bool *trackTransactions* ) [inline]
- 6.248.2.37 Pointer<Tracked> activemq::state::ConnectionStateTracker::track ( const Pointer< Command > & *command* ) throw ( decaf::io::IOException )
- 6.248.2.38 void activemq::state::ConnectionStateTracker::trackBack ( const Pointer< Command > & *command* )
- 6.248.2.39 void activemq::state::ConnectionStateTracker::transportInterrupted ( )

### 6.248.3 Friends And Related Function Documentation

- 6.248.3.1 friend class RemoveTransactionAction [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/state/ConnectionStateTracker.h

## 6.249 decaf::util::logging::ConsoleHandler Class Reference

This **Handler** (p. 1941) publishes log records to System.err.

```
#include <src/main/decaf/util/logging/ConsoleHandler.h>
```

Inheritance diagram for decaf::util::logging::ConsoleHandler:

### Public Member Functions

- **ConsoleHandler** ()
- virtual ~**ConsoleHandler** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Close the current output stream.*
- virtual void **publish** (const **LogRecord** &record)  
*Publish the Log Record to this **Handler** (p. 1941).*

### 6.249.1 Detailed Description

This **Handler** (p. 1941) publishes log records to System.err.

By default the **SimpleFormatter** (p. 3442) is used to generate brief summaries.

Configuration: By default each **ConsoleHandler** (p. 1367) is initialized using the following **LogManager** (p. 2363) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

ConsoleHandler.level specifies the default level for the **Handler** (p. 1941) (defaults to **Level.INFO** (p. 2295)). ConsoleHandler.filter specifies the name of a **Filter** (p. 1853) class to use (defaults to no **Filter** (p. 1853)). ConsoleHandler.formatter specifies the name of a **Formatter** (p. 1927) class to use (defaults to **SimpleFormatter** (p. 3442)).

#### Since

1.0

### 6.249.2 Constructor & Destructor Documentation

6.249.2.1 `decaf::util::logging::ConsoleHandler::ConsoleHandler ( )`

6.249.2.2 `virtual decaf::util::logging::ConsoleHandler::~~ConsoleHandler ( ) [inline, virtual]`

### 6.249.3 Member Function Documentation

6.249.3.1 `virtual void decaf::util::logging::ConsoleHandler::close ( ) throw ( decaf::io::IOException ) [virtual]`

Close the current output stream.

Override the **StreamHandler** (p. 3591) close to flush the Std Err stream but doesn't close.

#### Exceptions

<i>IOException</i>	
--------------------	--

Reimplemented from **decaf::util::logging::StreamHandler** (p. 3593).

6.249.3.2 `virtual void decaf::util::logging::ConsoleHandler::publish ( const LogRecord & record ) [virtual]`

Publish the Log Record to this **Handler** (p. 1941).

#### Parameters

<i>record</i>	The <b>LogRecord</b> (p. 2370) to Publish
---------------	---

Reimplemented from `decaf::util::logging::StreamHandler` (p. 3594).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ConsoleHandler.h`

## 6.250 `activemq::commands::ConsumerControl` Class Reference

```
#include <src/main/activemq/commands/ConsumerControl.h>
```

Inheritance diagram for `activemq::commands::ConsumerControl`:

### Public Member Functions

- **ConsumerControl** ()
- virtual **~ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConsumerControl** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int prefetch)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool flush)
- virtual bool **isStart** () const

- virtual void **setStart** (bool **start**)
- virtual bool **isStop** () const
- virtual void **setStop** (bool **stop**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONSUMERCONTROL** = 17

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- bool **close**
- **Pointer**< **ConsumerId** > **consumerId**
- int **prefetch**
- bool **flush**
- bool **start**
- bool **stop**

## 6.250.1 Constructor & Destructor Documentation

6.250.1.1 **activemq::commands::ConsumerControl::ConsumerControl** ( )

6.250.1.2 **virtual activemq::commands::ConsumerControl::~~ConsumerControl** ( )  
[virtual]

## 6.250.2 Member Function Documentation

6.250.2.1 **virtual ConsumerControl\*** **activemq::commands::ConsumerControl::cloneDataStructure**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.250.2.2 `virtual void activemq::commands::ConsumerControl::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.250.2.3 `virtual bool activemq::commands::ConsumerControl::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.250.2.4 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ( ) const [virtual]`

6.250.2.5 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId ( ) [virtual]`

6.250.2.6 `virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.250.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination ( ) const [virtual]`

- 6.250.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerControl::getDestination ( ) [virtual]`
- 6.250.2.9 `virtual int activemq::commands::ConsumerControl::getPrefetch ( ) const [virtual]`
- 6.250.2.10 `virtual bool activemq::commands::ConsumerControl::isClose ( ) const [virtual]`
- 6.250.2.11 `virtual bool activemq::commands::ConsumerControl::isFlush ( ) const [virtual]`
- 6.250.2.12 `virtual bool activemq::commands::ConsumerControl::isStart ( ) const [virtual]`
- 6.250.2.13 `virtual bool activemq::commands::ConsumerControl::isStop ( ) const [virtual]`
- 6.250.2.14 `virtual void activemq::commands::ConsumerControl::setClose ( bool close ) [virtual]`
- 6.250.2.15 `virtual void activemq::commands::ConsumerControl::setConsumerId ( const Pointer< ConsumerId > & consumerId ) [virtual]`
- 6.250.2.16 `virtual void activemq::commands::ConsumerControl::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.250.2.17 `virtual void activemq::commands::ConsumerControl::setFlush ( bool flush ) [virtual]`
- 6.250.2.18 `virtual void activemq::commands::ConsumerControl::setPrefetch ( int prefetch ) [virtual]`
- 6.250.2.19 `virtual void activemq::commands::ConsumerControl::setStart ( bool start ) [virtual]`
- 6.250.2.20 `virtual void activemq::commands::ConsumerControl::setStop ( bool stop ) [virtual]`
- 6.250.2.21 `virtual std::string activemq::commands::ConsumerControl::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

## 6.251 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller

### Class Reference

1377

6.250.2.22 `virtual Pointer<Command> activemq::commands::ConsumerControl::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.250.3 Field Documentation

6.250.3.1 `bool activemq::commands::ConsumerControl::close` [protected]

6.250.3.2 `Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId` [protected]

6.250.3.3 `Pointer<ActiveMQDestination> activemq::commands::ConsumerControl::destination` [protected]

6.250.3.4 `bool activemq::commands::ConsumerControl::flush` [protected]

6.250.3.5 `const unsigned char activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17` [static]

6.250.3.6 `int activemq::commands::ConsumerControl::prefetch` [protected]

6.250.3.7 `bool activemq::commands::ConsumerControl::start` [protected]

6.250.3.8 `bool activemq::commands::ConsumerControl::stop` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerControl.h`

## 6.251 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1373).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller`:

## Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.251.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1373).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

### 6.251.2 Constructor & Destructor Documentation

- 6.251.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::ConsumerControlMarshaller`  
( ) [inline]



## 6.251 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller

### Class Reference 1379

6.251.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::~~ConsumerControlMarshaller ( ) [inline, virtual]`

### 6.251.3 Member Function Documentation

6.251.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.251.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.251.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```

6.251.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```

6.251.3.5  virtual int activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

## 6.251 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller

### Class Reference 1381

6.251.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
(p. 768).

6.251.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
(p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerControlMarshaller.h**

## 6.252 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1378).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.252.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1378).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.252.2 Constructor & Destructor Documentation

6.252.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::ConsumerControlMarshaller ( ) [inline]`

6.252.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::~~ConsumerControlMarshaller ( ) [inline, virtual]`

## 6.252.3 Member Function Documentation

6.252.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.252.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.252.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.252.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.252.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.252 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller

### Class Reference 1385

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.252.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.252.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerControlMarshaller.h**

## 6.253 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1382).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.253.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1382).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.253.2 Constructor & Destructor Documentation

6.253.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::ConsumerControlMarshaller ( ) [inline]`

6.253.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::~~ConsumerControlMarshaller ( ) [inline, virtual]`

## 6.253.3 Member Function Documentation

6.253.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.253.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.253.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.253.3.4  virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.253.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.253 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller

### Class Reference 1389

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.253.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.253.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerControlMarshaller.h**

## 6.254 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1386).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.254.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1386).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.254.2 Constructor & Destructor Documentation

6.254.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::ConsumerControlMarshaller ( ) [inline]`

6.254.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::~~ConsumerControlMarshaller ( ) [inline, virtual]`

## 6.254.3 Member Function Documentation

6.254.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.254.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.254.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.254.3.4  virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.254.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.254 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller

### Class Reference 1393

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.254.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.254.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerControlMarshaller.h**

## 6.255 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1390).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.255.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1390).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.255.2 Constructor & Destructor Documentation

6.255.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::ConsumerControlMarshaller ( ) [inline]`

6.255.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::~~ConsumerControlMarshaller ( ) [inline, virtual]`

## 6.255.3 Member Function Documentation

6.255.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.255.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.255.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.255.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.255.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.255 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller

### Class Reference 1397

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.255.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.255.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConsumerControlMarshaller.h**

## 6.256 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1394).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller:

### Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.256.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1394).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.256.2 Constructor & Destructor Documentation

6.256.2.1 `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::ConsumerControlMarshaller ( ) [inline]`

6.256.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::~~ConsumerControlMarshaller ( ) [inline, virtual]`

## 6.256.3 Member Function Documentation

6.256.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.256.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.256.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

```
6.256.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.256.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.256 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller

### Class Reference 1401

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.256.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.256.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConsumerControlMarshaller.h**

## 6.257 activemq::commands::ConsumerId Class Reference

```
#include <src/main/activemq/commands/ConsumerId.h>
```

Inheritance diagram for activemq::commands::ConsumerId:

### Public Types

- typedef **decaf::lang::PointerComparator**< **ConsumerId** > **COMPARATOR**

### Public Member Functions

- **ConsumerId** ()
- **ConsumerId** (const **ConsumerId** &other)
- **ConsumerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConsumerId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **ConsumerId** &value) const
- virtual bool **equals** (const **ConsumerId** &value) const
- virtual bool **operator==** (const **ConsumerId** &value) const
- virtual bool **operator<** (const **ConsumerId** &value) const
- **ConsumerId** & **operator=** (const **ConsumerId** &other)



## Static Public Attributes

- static const unsigned char **ID\_CONSUMERID** = 122

## Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

### 6.257.1 Member Typedef Documentation

6.257.1.1 `typedef decaf::lang::PointerComparator<ConsumerId>  
activemq::commands::ConsumerId::COMPARATOR`

### 6.257.2 Constructor & Destructor Documentation

6.257.2.1 `activemq::commands::ConsumerId::ConsumerId ( )`

6.257.2.2 `activemq::commands::ConsumerId::ConsumerId ( const ConsumerId & other )`

6.257.2.3 `activemq::commands::ConsumerId::ConsumerId ( const SessionId & sessionId,  
long long consumerId )`

6.257.2.4 `virtual activemq::commands::ConsumerId::~~ConsumerId ( ) [virtual]`

### 6.257.3 Member Function Documentation

6.257.3.1 `virtual ConsumerId* activemq::commands::ConsumerId::cloneDataStructure ( )  
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1628).

6.257.3.2 `virtual int activemq::commands::ConsumerId::compareTo ( const ConsumerId &  
value ) const [virtual]`

6.257.3.3 `virtual void activemq::commands::ConsumerId::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.257.3.4 `virtual bool activemq::commands::ConsumerId::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.257.3.5 `virtual bool activemq::commands::ConsumerId::equals ( const ConsumerId & value ) const [virtual]`

6.257.3.6 `virtual std::string& activemq::commands::ConsumerId::getConnectionId ( ) [virtual]`

6.257.3.7 `virtual const std::string& activemq::commands::ConsumerId::getConnectionId ( ) const [virtual]`

6.257.3.8 `virtual unsigned char activemq::commands::ConsumerId::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.257.3.9 `const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId ( ) const`

- 6.257.3.10 virtual long long activemq::commands::ConsumerId::getSessionId ( ) const  
[virtual]
- 6.257.3.11 virtual long long activemq::commands::ConsumerId::getValue ( ) const  
[virtual]
- 6.257.3.12 virtual bool activemq::commands::ConsumerId::operator< ( const ConsumerId & value ) const [virtual]
- 6.257.3.13 ConsumerId& activemq::commands::ConsumerId::operator= ( const ConsumerId & other )
- 6.257.3.14 virtual bool activemq::commands::ConsumerId::operator== ( const ConsumerId & value ) const [virtual]
- 6.257.3.15 virtual void activemq::commands::ConsumerId::setConnectionId ( const std::string & connectionId ) [virtual]
- 6.257.3.16 virtual void activemq::commands::ConsumerId::setSessionId ( long long sessionId ) [virtual]
- 6.257.3.17 virtual void activemq::commands::ConsumerId::setValue ( long long value ) [virtual]
- 6.257.3.18 virtual std::string activemq::commands::ConsumerId::toString ( ) const  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

#### 6.257.4 Field Documentation

- 6.257.4.1 std::string activemq::commands::ConsumerId::connectionId  
[protected]
- 6.257.4.2 const unsigned char activemq::commands::ConsumerId::ID\_ -  
CONSUMERID = 122 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.257.4.3 `long long activemq::commands::ConsumerId::sessionId`  
`[protected]`

6.257.4.4 `long long activemq::commands::ConsumerId::value` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerId.h`

## 6.258 `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1402).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller`:

### Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.258.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1402).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.258.2 Constructor & Destructor Documentation

6.258.2.1 **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.258.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.258.3 Member Function Documentation

6.258.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.258.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.258.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.258.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.258.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.258.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.258.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerIdMarshaller.h**

## 6.259 **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1406).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**:

**Public Member Functions**

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*



- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.259.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1406).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.259.2 Constructor & Destructor Documentation

6.259.2.1 **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.259.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.259.3 Member Function Documentation

6.259.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.259.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.259.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.259.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.259.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.259.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.259.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerIdMarshaller.h**

## 6.260 **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1410).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**:

**Public Member Functions**

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.260.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1410).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.260.2 Constructor & Destructor Documentation

6.260.2.1 **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.260.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.260.3 Member Function Documentation

6.260.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.260.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.260.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.260.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.260.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.260.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.260.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerIdMarshaller.h**

## 6.261 **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1414).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**:

**Public Member Functions**

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*



- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.261.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1414).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.261.2 Constructor & Destructor Documentation

6.261.2.1 **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.261.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.261.3 Member Function Documentation

6.261.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.261.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.261.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.261.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.261.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.261.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.261.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerIdMarshaller.h**

## 6.262 **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1418).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**:

**Public Member Functions**

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.262.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1418).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.262.2 Constructor & Destructor Documentation

6.262.2.1 **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.262.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.262.3 Member Function Documentation

6.262.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.262.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```

6.262.3.3  virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```

6.262.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```

6.262.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.262.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.262.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConsumerIdMarshaller.h**

## 6.263 **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1422).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarsh
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller**:

**Public Member Functions**

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*



- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.263.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1422).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.263.2 Constructor & Destructor Documentation

6.263.2.1 **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::ConsumerIdMarshaller**  
( ) [inline]

6.263.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::~~ConsumerIdMarshaller**  
( ) [inline, virtual]

### 6.263.3 Member Function Documentation

6.263.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.263.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.263.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.263.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.263.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.263.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.263.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i> if an error occurs during the unmarshal.
---

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConsumerIdMarshaller.h**

**6.264 activemq::commands::ConsumerInfo Class Reference**

```
#include <src/main/activemq/commands/ConsumerInfo.h>
```

Inheritance diagram for **activemq::commands::ConsumerInfo**:

**Public Member Functions**

- **ConsumerInfo** ()
- virtual **~ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ConsumerInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- **Pointer< RemoveInfo > createRemoveCommand** () const
- virtual const **Pointer< ConsumerId > & getConsumerId** () const
- virtual **Pointer< ConsumerId > & getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer< ConsumerId > & consumerId**)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool browser)
- virtual const **Pointer< ActiveMQDestination > & getDestination** () const
- virtual **Pointer< ActiveMQDestination > & getDestination** ()
- virtual void **setDestination** (const **Pointer< ActiveMQDestination > & destination**)

- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int **prefetchSize**)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int **maximumPendingMessageLimit**)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool **dispatchAsync**)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &**selector**)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &**subscriptionName**)
- virtual bool **isNoLocal** () const
- virtual void **setNoLocal** (bool **noLocal**)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool **retroactive**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual const **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** () const
- virtual **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > &**additionalPredicate**)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool **networkSubscription**)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool **optimizedAcknowledge**)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool **noRangeAcks**)
- virtual const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** ()
- virtual void **setNetworkConsumerPath** (const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > &**networkConsumerPath**)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \***visitor**)  
throw ( **exceptions::ActiveMQException** )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONSUMERINFO** = 5

### Protected Attributes

- **Pointer< ConsumerId > consumerId**
- bool **browser**
- **Pointer< ActiveMQDestination > destination**
- int **prefetchSize**
- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**
- bool **noLocal**
- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**
- **Pointer< BooleanExpression > additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**
- std::vector< **decaf::lang::Pointer< ConsumerId > > networkConsumerPath**

## 6.264.1 Constructor & Destructor Documentation

6.264.1.1 **activemq::commands::ConsumerInfo::ConsumerInfo ( )**

6.264.1.2 **virtual activemq::commands::ConsumerInfo::~~ConsumerInfo ( ) [virtual]**

## 6.264.2 Member Function Documentation

6.264.2.1 **virtual ConsumerInfo\* activemq::commands::ConsumerInfo::cloneDataStructure ( ) const [virtual]**

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.264.2.2 `virtual void activemq::commands::ConsumerInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code><i>src</i></code>	- Source Object
-------------------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 724).

6.264.2.3 `Pointer<RemoveInfo> activemq::commands::ConsumerInfo::createRemoveCommand ( ) const`

6.264.2.4 `virtual bool activemq::commands::ConsumerInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 725).

6.264.2.5 `virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate ( ) const [virtual]`

6.264.2.6 `virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate ( ) [virtual]`

6.264.2.7 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath ( ) const [virtual]`

6.264.2.8 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath ( ) [virtual]`

6.264.2.9 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId ( ) [virtual]`

6.264.2.10 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId ( ) const`  
[virtual]

6.264.2.11 `virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.264.2.12 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination ( ) const` [virtual]

6.264.2.13 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination ( )` [virtual]

6.264.2.14 `virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit ( ) const` [virtual]

6.264.2.15 `virtual const std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath ( ) const`  
[virtual]

6.264.2.16 `virtual std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath ( )`  
[virtual]

6.264.2.17 `virtual int activemq::commands::ConsumerInfo::getPrefetchSize ( ) const`  
[virtual]

6.264.2.18 `virtual unsigned char activemq::commands::ConsumerInfo::getPriority ( ) const`  
[virtual]

6.264.2.19 `virtual const std::string& activemq::commands::ConsumerInfo::getSelector ( ) const` [virtual]

6.264.2.20 `virtual std::string& activemq::commands::ConsumerInfo::getSelector ( )`  
[virtual]

6.264.2.21 `virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName ( ) const` [virtual]



6.264.2.22 `virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName ( )`  
[virtual]

6.264.2.23 `virtual bool activemq::commands::ConsumerInfo::isBrowser ( ) const`  
[virtual]

6.264.2.24 `virtual bool activemq::commands::ConsumerInfo::isConsumerInfo ( ) const`  
[inline, virtual]

### Returns

an answer of true to the `isConsumerInfo()` (p. 1431) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 726).

6.264.2.25 `virtual bool activemq::commands::ConsumerInfo::isDispatchAsync ( ) const`  
[virtual]

6.264.2.26 `virtual bool activemq::commands::ConsumerInfo::isExclusive ( ) const`  
[virtual]

6.264.2.27 `virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription ( ) const`  
[virtual]

6.264.2.28 `virtual bool activemq::commands::ConsumerInfo::isNoLocal ( ) const`  
[virtual]

6.264.2.29 `virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks ( ) const`  
[virtual]

6.264.2.30 `virtual bool activemq::commands::ConsumerInfo::isOptimizedAcknowledge ( )`  
`const [virtual]`

6.264.2.31 `virtual bool activemq::commands::ConsumerInfo::isRetroactive ( ) const`  
[virtual]

6.264.2.32 `virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate ( const`  
`Pointer< BooleanExpression > & additionalPredicate ) [virtual]`

6.264.2.33 `virtual void activemq::commands::ConsumerInfo::setBrokerPath ( const`  
`std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath )`  
[virtual]

6.264.2.34 `virtual void activemq::commands::ConsumerInfo::setBrowser ( bool browser )`  
[virtual]

6.264.2.35 `virtual void activemq::commands::ConsumerInfo::setConsumerId ( const Pointer<`  
`ConsumerId > & consumerId ) [virtual]`

- 6.264.2.36 `virtual void activemq::commands::ConsumerInfo::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.264.2.37 `virtual void activemq::commands::ConsumerInfo::setDispatchAsync ( bool dispatchAsync ) [virtual]`
- 6.264.2.38 `virtual void activemq::commands::ConsumerInfo::setExclusive ( bool exclusive ) [virtual]`
- 6.264.2.39 `virtual void activemq::commands::ConsumerInfo::setMaximumPendingMessageLimit ( int maximumPendingMessageLimit ) [virtual]`
- 6.264.2.40 `virtual void activemq::commands::ConsumerInfo::setNetworkConsumerPath ( const std::vector< decaf::lang::Pointer< ConsumerId > > & networkConsumerPath ) [virtual]`
- 6.264.2.41 `virtual void activemq::commands::ConsumerInfo::setNetworkSubscription ( bool networkSubscription ) [virtual]`
- 6.264.2.42 `virtual void activemq::commands::ConsumerInfo::setNoLocal ( bool noLocal ) [virtual]`
- 6.264.2.43 `virtual void activemq::commands::ConsumerInfo::setNoRangeAcks ( bool noRangeAcks ) [virtual]`
- 6.264.2.44 `virtual void activemq::commands::ConsumerInfo::setOptimizedAcknowledge ( bool optimizedAcknowledge ) [virtual]`
- 6.264.2.45 `virtual void activemq::commands::ConsumerInfo::setPrefetchSize ( int prefetchSize ) [virtual]`
- 6.264.2.46 `virtual void activemq::commands::ConsumerInfo::setPriority ( unsigned char priority ) [virtual]`
- 6.264.2.47 `virtual void activemq::commands::ConsumerInfo::setRetroactive ( bool retroactive ) [virtual]`
- 6.264.2.48 `virtual void activemq::commands::ConsumerInfo::setSelector ( const std::string & selector ) [virtual]`
- 6.264.2.49 `virtual void activemq::commands::ConsumerInfo::setSubscriptionName ( const std::string & subscriptionName ) [virtual]`
- 6.264.2.50 `virtual std::string activemq::commands::ConsumerInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

**6.264.2.51** `virtual Pointer<Command> activemq::commands::ConsumerInfo::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

**6.264.3 Field Documentation**

**6.264.3.1** `Pointer<BooleanExpression> ac-  
tivemq::commands::ConsumerInfo::additionalPredicate  
[protected]`

**6.264.3.2** `std::vector< decaf::lang::Pointer<BrokerId> >  
activemq::commands::ConsumerInfo::brokerPath [protected]`

**6.264.3.3** `bool activemq::commands::ConsumerInfo::browser [protected]`

**6.264.3.4** `Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId  
[protected]`

**6.264.3.5** `Pointer<ActiveMQDestination> ac-  
tivemq::commands::ConsumerInfo::destination  
[protected]`

**6.264.3.6** `bool activemq::commands::ConsumerInfo::dispatchAsync  
[protected]`

**6.264.3.7** `bool activemq::commands::ConsumerInfo::exclusive [protected]`

**6.264.3.8** `const unsigned char activemq::commands::ConsumerInfo::ID_-  
CONSUMERINFO = 5 [static]`

**6.264.3.9** `int activemq::commands::ConsumerInfo::maximumPendingMessageLimit  
[protected]`

- 6.264.3.10 `std::vector< decaf::lang::Pointer<ConsumerId> >`  
`activemq::commands::ConsumerInfo::networkConsumerPath`  
[protected]
- 6.264.3.11 `bool activemq::commands::ConsumerInfo::networkSubscription`  
[protected]
- 6.264.3.12 `bool activemq::commands::ConsumerInfo::noLocal` [protected]
- 6.264.3.13 `bool activemq::commands::ConsumerInfo::noRangeAcks`  
[protected]
- 6.264.3.14 `bool activemq::commands::ConsumerInfo::optimizedAcknowledge`  
[protected]
- 6.264.3.15 `int activemq::commands::ConsumerInfo::prefetchSize`  
[protected]
- 6.264.3.16 `unsigned char activemq::commands::ConsumerInfo::priority`  
[protected]
- 6.264.3.17 `bool activemq::commands::ConsumerInfo::retroactive`  
[protected]
- 6.264.3.18 `std::string activemq::commands::ConsumerInfo::selector`  
[protected]
- 6.264.3.19 `std::string activemq::commands::ConsumerInfo::subscriptionName`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerInfo.h`

## 6.265 `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1434).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMar
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller`:

## Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.265.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1434).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.265.2 Constructor & Destructor Documentation

6.265.2.1 **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::ConsumerInfoMarshaller**  
 ( ) [inline]

6.265.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller**  
 ( ) [inline, virtual]

### 6.265.3 Member Function Documentation

```
6.265.3.1 virtual commands::DataStructure* ac-
          tivemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::createObject (
          ) const [virtual]
```

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.265.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::getDataStructureType
          ( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.265.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseMarshal
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
            decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.265.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
 (p. 766).

6.265.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal1  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
 (p. 767).

6.265.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 768).

6.265.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 769).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h`



## 6.266 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1439).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.266.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1439).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.266.2 Constructor & Destructor Documentation

6.266.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]`

6.266.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller ( ) [inline, virtual]`

## 6.266.3 Member Function Documentation

6.266.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.266.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.266.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.266.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.266.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.266.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.266.3.7  virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerInfoMarshaller.h**

## 6.267 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1443).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.267.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1443).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.267.2 Constructor & Destructor Documentation

6.267.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]`

6.267.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller ( ) [inline, virtual]`

## 6.267.3 Member Function Documentation

6.267.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.267.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.267.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.267.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

6.267.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.267.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.267.3.7  virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ConsumerInfoMarshaller.h**



## 6.268 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1447).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.268.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1447).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.268.2 Constructor & Destructor Documentation

6.268.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]`

6.268.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller ( ) [inline, virtual]`

## 6.268.3 Member Function Documentation

6.268.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.268.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.268.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.268.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

6.268.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.268.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.268.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerInfoMarshaller.h**

## 6.269 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1451).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.269.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1451).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.269.2 Constructor & Destructor Documentation

6.269.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]`

6.269.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller ( ) [inline, virtual]`

## 6.269.3 Member Function Documentation

6.269.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.269.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.269.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.269.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

6.269.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.269.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.269.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ConsumerInfoMarshaller.h**



## 6.270 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1455).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller:

### Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.270.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1455).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.270.2 Constructor & Destructor Documentation

6.270.2.1 `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::ConsumerInfoMarshaller ( ) [inline]`

6.270.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller ( ) [inline, virtual]`

## 6.270.3 Member Function Documentation

6.270.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.270.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.270.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.270.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.270.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.270.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.270.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ConsumerInfoMarshaller.h**

## 6.271 activemq::state::ConsumerState Class Reference

```
#include <src/main/activemq/state/ConsumerState.h>
```

### Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual **~ConsumerState** ()
- std::string **toString** () const
- const **Pointer**< **ConsumerInfo** > &**getInfo** () const

### 6.271.1 Constructor & Destructor Documentation

6.271.1.1 **activemq::state::ConsumerState::ConsumerState** ( const **Pointer**< **ConsumerInfo** > & *info* )

6.271.1.2 virtual **activemq::state::ConsumerState::~~ConsumerState** ( ) [virtual]

### 6.271.2 Member Function Documentation

6.271.2.1 const **Pointer**<**ConsumerInfo**>& **activemq::state::ConsumerState::getInfo** ( ) const [inline]

6.271.2.2 std::string **activemq::state::ConsumerState::toString** ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConsumerState.h**

## 6.272 activemq::commands::ControlCommand Class Reference

```
#include <src/main/activemq/commands/ControlCommand.h>
```

Inheritance diagram for **activemq::commands::ControlCommand**:

### Public Member Functions

- **ControlCommand** ()
- virtual **~ControlCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ControlCommand** \* **cloneDataStructure** () const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getCommand** () const
- virtual std::string & **getCommand** ()
- virtual void **setCommand** (const std::string &command)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_CONTROLCOMMAND** = 14

### Protected Attributes

- std::string **command**

## 6.272.1 Constructor & Destructor Documentation

6.272.1.1 **activemq::commands::ControlCommand::ControlCommand ( )**

6.272.1.2 **virtual activemq::commands::ControlCommand::~~ControlCommand ( )**  
[virtual]

## 6.272.2 Member Function Documentation

6.272.2.1 **virtual ControlCommand\* activemq::commands::ControlCommand::cloneDataStructure ( ) const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.272.2.2 `virtual void activemq::commands::ControlCommand::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.272.2.3 `virtual bool activemq::commands::ControlCommand::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.272.2.4 `virtual std::string& activemq::commands::ControlCommand::getCommand ( ) [virtual]`

6.272.2.5 `virtual const std::string& activemq::commands::ControlCommand::getCommand ( ) const [virtual]`

6.272.2.6 `virtual unsigned char activemq::commands::ControlCommand::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.272.2.7 `virtual void activemq::commands::ControlCommand::setCommand ( const std::string & command ) [virtual]`

6.272.2.8 `virtual std::string activemq::commands::ControlCommand::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.272.2.9 `virtual Pointer<Command> activemq::commands::ControlCommand::visit`  
`( activemq::state::CommandVisitor * visitor ) throw (`  
`exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.272.3 Field Documentation

6.272.3.1 `std::string activemq::commands::ControlCommand::command`  
`[protected]`

6.272.3.2 `const unsigned char activemq::commands::ControlCommand::ID_ -`  
`CONTROLCOMMAND = 14 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ControlCommand.h`

## 6.273 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1462).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandM
```



Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller:

## Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.273.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1462).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.273.2 Constructor & Destructor Documentation

6.273.2.1 **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::ControlCommandMarshaller**  
 ( ) [inline]

6.273.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::~ControlCommandMarshaller ( ) [inline, virtual]`

### 6.273.3 Member Function Documentation

6.273.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.273.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.273.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

## 6.273 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller

### Class Reference 1469

6.273.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

6.273.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

6.273.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 768).

6.273.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 769).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h`

## 6.274 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1467).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.274.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1467).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.274.2 Constructor & Destructor Documentation

6.274.2.1 `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::ControlCommandMarshaller ( ) [inline]`

6.274.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::~~ControlCommandMarshaller ( ) [inline, virtual]`

## 6.274.3 Member Function Documentation

6.274.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.274.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.274.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.274.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.274.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.274.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.274.3.7  virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ControlCommandMarshaller.h**



## 6.275 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1471).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.275.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1471).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.275.2 Constructor & Destructor Documentation

6.275.2.1 `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::ControlCommandMarshaller ( ) [inline]`

6.275.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::~~ControlCommandMarshaller ( ) [inline, virtual]`

## 6.275.3 Member Function Documentation

6.275.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.275.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.275.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.275.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

6.275.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.275.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.275.3.7  virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ControlCommandMarshaller.h**

## 6.276 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1475).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.276.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1475).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.276.2 Constructor & Destructor Documentation

6.276.2.1 `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::ControlCommandMarshaller ( ) [inline]`

6.276.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::~~ControlCommandMarshaller ( ) [inline, virtual]`

## 6.276.3 Member Function Documentation

6.276.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.276.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.276.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.276.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

6.276.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.276.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.276.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ControlCommandMarshaller.h**



## 6.277 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1479).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.277.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1479).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.277.2 Constructor & Destructor Documentation

6.277.2.1 `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::ControlCommandMarshaller ( ) [inline]`

6.277.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::~~ControlCommandMarshaller ( ) [inline, virtual]`

## 6.277.3 Member Function Documentation

6.277.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.277.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.277.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.277.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
[virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

6.277.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal1 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.277.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.277.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ControlCommandMarshaller.h**

## 6.278 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1483).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller:

### Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.278.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1483).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.278.2 Constructor & Destructor Documentation

6.278.2.1 `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::ControlCommandMarshaller ( ) [inline]`

6.278.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::~~ControlCommandMarshaller ( ) [inline, virtual]`

## 6.278.3 Member Function Documentation

6.278.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.278.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.278.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.278.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.278.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.278.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.278.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ControlCommandMarshaller.h**



## 6.279 decaf::util::concurrent::CountDownLatch Class Reference

```
#include <src/main/decaf/util/concurrent/CountDownLatch.h>
```

### Public Member Functions

- **CountDownLatch** (int count)  
*Constructor.*
- virtual **~CountDownLatch** ()
- virtual void **await** () throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )  
*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.*
- virtual bool **await** (long long timeout) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )  
*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.*
- virtual bool **await** (long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )  
*Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.*
- virtual void **countDown** ()  
*Counts down the latch, releasing all waiting threads when the count hits zero.*
- virtual int **getCount** () const  
*Gets the current count.*

### 6.279.1 Constructor & Destructor Documentation

#### 6.279.1.1 decaf::util::concurrent::CountDownLatch::CountDownLatch ( int count )

Constructor.

#### Parameters

<i>count</i>	- number to count down from.
--------------	------------------------------

#### 6.279.1.2 virtual decaf::util::concurrent::CountDownLatch::~~CountDownLatch ( ) [virtual]

### 6.279.2 Member Function Documentation

6.279.2.1 `virtual void decaf::util::concurrent::CountDownLatch::await ( ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )`  
`[virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted.

If the current count is zero then this method returns immediately.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happen:

- \* The count reaches zero due to invocations of the **countDown()** (p. 1489) method; or
- \* Some other thread interrupts the current thread.

If the current thread:

- \* has its interrupted status set on entry to this method; or \*
  - is interrupted while waiting,
- then InterruptedException is thrown and the current thread's interrupted status is cleared.

#### Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

6.279.2.2 `virtual bool decaf::util::concurrent::CountDownLatch::await ( long long timeOut ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )` `[virtual]`

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

- \* The count reaches zero due to invocations of the **countDown()** (p. 1489) method; or \*
- Some other thread interrupts the current thread; or \*
- The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

- \* has its interrupted status set on entry to this method; or \*
  - is interrupted while waiting,
- then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Parameters

<i>timeout</i>	- Time in milliseconds to wait for the count to reach zero.
----------------	---

**Exceptions**

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

**6.279.2.3** `virtual bool decaf::util::concurrent::CountDownLatch::await ( long long timeout, const TimeUnit & unit ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::Exception )` [virtual]

Causes the current thread to wait until the latch has counted down to zero, unless the thread is interrupted, or the specified waiting time elapses.

If the current count is zero then this method returns immediately with the value true.

If the current count is greater than zero then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happen:

\* The count reaches zero due to invocations of the **countDown()** (p. 1489) method; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the count reaches zero then the method returns with the value true.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting, then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

**Parameters**

<i>timeout</i>	- Time to wait for the count to reach zero.
<i>unit</i>	- The units that the timeout specifies.

**Exceptions**

<i>InterruptedException</i>	- if the current thread is interrupted while waiting.
<i>Exception</i>	- if any other error occurs.

**6.279.2.4** `virtual void decaf::util::concurrent::CountDownLatch::countDown ( )` [virtual]

Counts down the latch, releasing all waiting threads when the count hits zero.

**6.279.2.5** `virtual int decaf::util::concurrent::CountDownLatch::getCount ( ) const` [inline, virtual]

Gets the current count.

**Returns**

int count value

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CountDownLatch.h**

**6.280 decaf::util::zip::CRC32 Class Reference**

Class that can be used to compute a CRC-32 checksum for a data stream.

```
#include <src/main/decaf/util/zip/CRC32.h>
```

Inheritance diagram for decaf::util::zip::CRC32:

**Public Member Functions**

- **CRC32** ()
- virtual **~CRC32** ()
- virtual long long **getValue** () const
- virtual void **reset** ()  
*Reset the checksum to its initial value.*
- virtual void **update** (const std::vector< unsigned char > &buffer)  
*Updates the current checksum with the specified vector of bytes.*
- virtual void **update** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Updates the current checksum with the specified array of bytes.*
- virtual void **update** (int byte)  
*Updates the current checksum with the specified byte value.*

**6.280.1 Detailed Description**

Class that can be used to compute a CRC-32 checksum for a data stream.

**Since**

1.0

## 6.280.2 Constructor & Destructor Documentation

6.280.2.1 `decaf::util::zip::CRC32 ( )`

6.280.2.2 `virtual decaf::util::zip::CRC32::~~CRC32 ( ) [virtual]`

## 6.280.3 Member Function Documentation

6.280.3.1 `virtual long long decaf::util::zip::CRC32::getValue ( ) const [virtual]`

### Returns

the current checksum value.

Implements **decaf::util::zip::Checksum** (p. 1115).

6.280.3.2 `virtual void decaf::util::zip::CRC32::reset ( ) [virtual]`

Reset the checksum to its initial value.

Implements **decaf::util::zip::Checksum** (p. 1115).

6.280.3.3 `virtual void decaf::util::zip::CRC32::update ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Updates the current checksum with the specified array of bytes.

### Parameters

<i>buffer</i>	The buffer to read the updated bytes from.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

### Exceptions

<i>NullPointerException</i>	if the passed buffer is NULL.
<i>IndexOutOfBoundsException</i>	if <code>offset + length &gt; size</code> of the buffer.

Implements **decaf::util::zip::Checksum** (p. 1115).

6.280.3.4 `virtual void decaf::util::zip::CRC32::update ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Updates the current checksum with the specified array of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
<i>offset</i>	The position in the buffer to start reading.
<i>length</i>	The amount of data to read from the byte buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if $\text{offset} + \text{length} > \text{size of the buffer}$ .
----------------------------------	--

Implements **decaf::util::zip::Checksum** (p. 1116).

6.280.3.5 `virtual void decaf::util::zip::CRC32::update ( const std::vector< unsigned char > & buffer ) [virtual]`

Updates the current checksum with the specified vector of bytes.

**Parameters**

<i>buffer</i>	The buffer to read the updated bytes from.
---------------	--

Implements **decaf::util::zip::Checksum** (p. 1116).

6.280.3.6 `virtual void decaf::util::zip::CRC32::update ( int byte ) [virtual]`

Updates the current checksum with the specified byte value.

**Parameters**

<i>byte</i>	The byte value to update the current <b>Checksum</b> (p. 1114) with (0..255).
-------------	---

Implements **decaf::util::zip::Checksum** (p. 1116).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/CRC32.h`

**6.281 ct\_data\_s Struct Reference**

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

**Data Fields**

- union {  
    **ush freq**  
    **ush code**

```

    } fc

    • union {
        ush dad
        ush len
    } dl

```

### 6.281.1 Field Documentation

6.281.1.1 ush ct\_data\_s::code

6.281.1.2 ush ct\_data\_s::dad

6.281.1.3 union { ... } ct\_data\_s::dl

6.281.1.4 union { ... } ct\_data\_s::fc

6.281.1.5 ush ct\_data\_s::freq

6.281.1.6 ush ct\_data\_s::len

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/deflate.h

## 6.282 activemq::commands::DataArrayResponse Class Reference

```
#include <src/main/activemq/commands/DataArrayResponse.h>
```

Inheritance diagram for activemq::commands::DataArrayResponse:

### Public Member Functions

- **DataArrayResponse** ()
- virtual **~DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataArrayResponse** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSet** \*value) const  
*Compares the **DataSet** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const std::vector< **decaf::lang::Pointer**< **DataSet** > > & **getData** () const
- virtual std::vector< **decaf::lang::Pointer**< **DataSet** > > & **getData** ()
- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataSet** > > &data)

### Static Public Attributes

- static const unsigned char **ID\_DATAARRAYRESPONSE** = 33

### Protected Attributes

- std::vector< **decaf::lang::Pointer**< **DataSet** > > **data**

## 6.282.1 Constructor & Destructor Documentation

6.282.1.1 **activemq::commands::DataArrayResponse::DataArrayResponse** ( )

6.282.1.2 **virtual activemq::commands::DataArrayResponse::~~DataArrayResponse** ( )  
[virtual]

## 6.282.2 Member Function Documentation

6.282.2.1 **virtual DataSet\*** **activemq::commands::DataArrayResponse::cloneDataSet** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 3228).

6.282.2.2 **virtual void** **activemq::commands::DataArrayResponse::copyDataSet** ( const **DataSet** \* src ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.



**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::Response** (p. 3229).

6.282.2.3 `virtual bool activemq::commands::DataArrayResponse::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 3229).

6.282.2.4 `virtual std::vector< decaf::lang::Pointer<DataStructure> > & activemq::commands::DataArrayResponse::getData ( ) [virtual]`

6.282.2.5 `virtual const std::vector< decaf::lang::Pointer<DataStructure> > & activemq::commands::DataArrayResponse::getData ( ) const [virtual]`

6.282.2.6 `virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Response** (p. 3230).

6.282.2.7 `virtual void activemq::commands::DataArrayResponse::setData ( const std::vector< decaf::lang::Pointer< DataStructure > > & data ) [virtual]`

6.282.2.8 `virtual std::string activemq::commands::DataArrayResponse::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3230).

### 6.282.3 Field Documentation

6.282.3.1 `std::vector< decaf::lang::Pointer<DataStructure> >`  
`activemq::commands::DataArrayResponse::data` [protected]

6.282.3.2 `const unsigned char activemq::commands::DataArrayResponse::ID_`  
`DATAARRAYRESPONSE = 33` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataArrayResponse.h`

## 6.283 `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1496).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller`:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

6.283

**activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

**Class Reference**

**1501**

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.283.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1496).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.283.2 Constructor & Destructor Documentation

6.283.2.1 **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.283.2.2 **virtual activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

### 6.283.3 Member Function Documentation

6.283.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3242).

6.283.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3242).

```
6.283.3.3 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.283.3.4 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.283.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

6.283

**activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

**Class Reference**

**1503**

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

```
6.283.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

```
6.283.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataArrayResponseMarshaller.h**

## 6.284 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1500).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**:

### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

## 6.284

### activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller

#### Class Reference

1505

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

#### 6.284.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1500).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.284.2 Constructor & Destructor Documentation

6.284.2.1 **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.284.2.2 **virtual activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

#### 6.284.3 Member Function Documentation

6.284.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

##### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3251).

6.284.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

##### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

```
6.284.3.3  virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

```
6.284.3.4  virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253).



6.284

activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller

## Class Reference

1507

```
6.284.3.5 virtual int activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal1  
    ( OpenWireFormat * wireFormat, commands::DataStructure *  
      dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
    [virtual]
```

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
(p. 3253).

```
6.284.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal2  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw  
    ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
(p. 3254).

```
6.284.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**

## 6.285 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1504).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**:

#### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

6.285

**activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

**Class Reference**

**1509**

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.285.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1504).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.285.2 Constructor & Destructor Documentation

6.285.2.1 **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.285.2.2 **virtual activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

## 6.285.3 Member Function Documentation

6.285.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3237).

```
6.285.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

```
6.285.3.3 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

```
6.285.3.4 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.285

**activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

**Class Reference**

1511

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

```
6.285.3.5  virtual int activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

```
6.285.3.6  virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

```
6.285.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataArrayResponseMarshaller.h**

## 6.286 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1508).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**:

#### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

6.286

**activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

**Class Reference**

**1513**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.286.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1508).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.286.2 Constructor & Destructor Documentation

6.286.2.1 **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.286.2.2 **virtual activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

## 6.286.3 Member Function Documentation

6.286.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

```
6.286.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

```
6.286.3.3 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.286.3.4 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



6.286

**activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

**Class Reference**

1515

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.286.3.5  virtual int activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

```
6.286.3.6  virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

```
6.286.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3259).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataArrayResponseMarshaller.h**

## 6.287 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1512).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**:

#### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

6.287

**activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

**Class Reference**

**1517**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.287.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1512).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.287.2 Constructor & Destructor Documentation

6.287.2.1 **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.287.2.2 **virtual activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

### 6.287.3 Member Function Documentation

6.287.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

```
6.287.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

```
6.287.3.3 virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

```
6.287.3.4 virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.287

**activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

**Class Reference**

1519

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

```
6.287.3.5  virtual int activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

```
6.287.3.6  virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3249).

```
6.287.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DataArrayResponseMarshaller.h**

## 6.288 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1516).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**:

#### Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

6.288

**activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**

**Class Reference**

**1521**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.288.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1516).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.288.2 Constructor & Destructor Documentation

6.288.2.1 **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::DataArrayResponseMarshaller**  
( ) [inline]

6.288.2.2 **virtual activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::~~DataArrayResponseMarshaller**  
( ) [inline, virtual]

## 6.288.3 Member Function Documentation

6.288.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

```
6.288.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

```
6.288.3.3  virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

```
6.288.3.4  virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



6.288

**activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**

**Class Reference**

1523

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262).

6.288.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262).

6.288.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3263).

```

6.288.3.7 virtual void activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3264).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DataArrayResponseMarshaller.h**

## 6.289 decaf::util::zip::DataFormatException Class Reference

```
#include <src/main/decaf/util/zip/DataFormatException.h>
```

Inheritance diagram for decaf::util::zip::DataFormatException:

#### Public Member Functions

- **DataFormatException** () throw ()  
*Default Constructor.*
- **DataFormatException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **DataFormatException** (const DataFormatException &ex) throw ()  
*Copy Constructor.*
- **DataFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **DataFormatException** (const std::exception \*cause) throw ()  
*Constructor.*

- **DataFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **DataFormatException** \* **clone** () const  
*Clones this exception.*
- virtual ~**DataFormatException** () throw ()

## 6.289.1 Constructor & Destructor Documentation

6.289.1.1 decaf::util::zip::DataFormatException::DataFormatException ( ) throw ()  
[inline]

Default Constructor.

6.289.1.2 decaf::util::zip::DataFormatException::DataFormatException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.289.1.3 decaf::util::zip::DataFormatException::DataFormatException ( const DataFormatException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.289.1.4 decaf::util::zip::DataFormatException::DataFormatException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.289.1.5 `decaf::util::zip::DataFormatException::DataFormatException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.289.1.6 `decaf::util::zip::DataFormatException::DataFormatException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.289.1.7 `virtual decaf::util::zip::DataFormatException::~DataFormatException ( ) throw () [inline, virtual]`

## 6.289.2 Member Function Documentation

6.289.2.1 `virtual DataFormatException* decaf::util::zip::DataFormatException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/DataFormatException.h`

## 6.290 decaf::io::DataInput Class Reference

The **DataInput** (p. 1523) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

```
#include <src/main/decaf/io/DataInput.h>
```

### Public Member Functions

- virtual **~DataInput** ()
- virtual bool **readBoolean** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.*
- virtual char **readByte** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads and returns one input byte.*
- virtual unsigned char **readUnsignedByte** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.*
- virtual char **readChar** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an input char and returns the char value.*
- virtual double **readDouble** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a double value.*
- virtual float **readFloat** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns a float value.*
- virtual int **readInt** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns an int value.*
- virtual long long **readLong** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a long value.*
- virtual short **readShort** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns a short value.*
- virtual unsigned short **readUnsignedShort** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns an int value in the range 0 through 65535.*
- virtual std::string **readString** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an NULL terminated ASCII string to the stream and returns the string to the caller.*
- virtual std::string **readLine** ()=0 throw ( decaf::io::IOException )

*Reads the next line of text from the input stream.*

- virtual std::string **readUTF** ()=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException )

*Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.*

- virtual void **readFully** (unsigned char \*buffer, int size)=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads some bytes from an input stream and stores them into the buffer array buffer.*

- virtual void **readFully** (unsigned char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Reads length bytes from an input stream.*

- virtual long long **skipBytes** (long long num)=0 throw ( io::IOException )

*Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.*

### 6.290.1 Detailed Description

The **DataInput** (p. 1523) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.

There is also a facility for reconstructing Strings from data in the Java standard modified UTF-8 format.

It is generally true of all the reading routines in this interface that if end of file is reached before the desired number of bytes has been read, an **EOFException** (p. 1789) is thrown. If any byte cannot be read for any reason other than end of file, an **IOException** (p. 2103) other than **EOFException** (p. 1789) is thrown. for example, an **IOException** (p. 2103) may be thrown if the underlying input stream has been closed.

#### See also

**DataOutput** (p. 1541)

**DataInputStream** (p. 1532)

#### Since

1.0

### 6.290.2 Constructor & Destructor Documentation

6.290.2.1 virtual decaf::io::DataInput::~DataInput ( ) [inline, virtual]

### 6.290.3 Member Function Documentation

6.290.3.1 virtual bool decaf::io::DataInput::readBoolean ( ) throw ( decaf::io::IOException,  
decaf::io::EOFException ) [pure virtual]

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

#### Returns

the boolean value of the read in byte (0=false, 1=true).

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.2 virtual char decaf::io::DataInput::readByte ( ) throw ( decaf::io::IOException,  
decaf::io::EOFException ) [pure virtual]

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

#### Returns

the 8-bit value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.3 virtual char decaf::io::DataInput::readChar ( ) throw ( decaf::io::IOException,  
decaf::io::EOFException ) [pure virtual]

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

#### Returns

the 8 bit char read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.4 `virtual double decaf::io::DataInput::readDouble ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the readlong method, then converting this long value to a double in exactly the manner of the method Double::longBitsToDouble.

#### Returns

the double value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.5 `virtual float decaf::io::DataInput::readFloat ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float::intBitsToFloat.

#### Returns

the float value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.6 `virtual void decaf::io::DataInput::readFully ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: \* length bytes of input data are available, in which case a normal return is made. \* End of file is detected, in



which case an **EOFException** (p. 1789) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2103) other than **EOFException** (p. 1789) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[offset], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to length.

#### Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.
<i>offset</i>	The location in buffer to start writing.
<i>length</i>	The number of bytes to read from the buffer.

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O Error occurs.
<b>EOFException</b> (p. 1789)	if the end of input is reached.
<b>NullPointerException</b>	if the buffer is NULL.
<b>IndexOutOfBoundsException</b>	if the offset + length > size, or an int param is negative.

```
6.290.3.7 virtual void decaf::io::DataInput::readFully ( unsigned char * buffer, int
               size ) throw ( decaf::io::IOException, decaf::io::EOFException,
               decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
               virtual]
```

Reads some bytes from an input stream and stores them into the buffer array buffer.

The number of bytes read is equal to the length of buffer.

This method blocks until one of the following conditions occurs: \* buffer's size bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1789) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2103) other than **EOFException** (p. 1789) is thrown.

If buffer size is zero, then no bytes are read. Otherwise, the first byte read is stored into element b[0], the next one into buffer[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of buffer have been updated with data from the input stream.

#### Parameters

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.
<b><i>IndexOutOfBoundsException</i></b>	if the size value is negative.

**6.290.3.8** `virtual int decaf::io::DataInput::readInt ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads four input bytes and returns an int value.

Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$$(((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff))$$
**Returns**

the int value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

**6.290.3.9** `virtual std::string decaf::io::DataInput::readLine ( ) throw ( decaf::io::IOException )` [pure virtual]

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character '

' is encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character '

', then that is discarded also; reading then ceases. If end of file is encountered before either of the characters '

' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

**Returns**

the next line of text read from the input stream or empty string if at EOF.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
--	-------------------------

6.290.3.10 `virtual long long decaf::io::DataInput::readLong ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]`

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

$$(((\text{long})(a \& 0\text{xff}) << 56) | ((\text{long})(b \& 0\text{xff}) << 48) | ((\text{long})(c \& 0\text{xff}) << 40) | ((\text{long})(d \& 0\text{xff}) << 32) | ((\text{long})(e \& 0\text{xff}) << 24) | ((\text{long})(f \& 0\text{xff}) << 16) | ((\text{long})(g \& 0\text{xff}) << 8) | ((\text{long})(h \& 0\text{xff})))$$
**Returns**

the 64 bit long long read.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.11 `virtual short decaf::io::DataInput::readShort ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [pure virtual]`

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

$$(\text{short})((a << 8) | (b \& 0\text{xff}))$$
**Returns**

the 16 bit short value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
--	-------------------------

<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.
---	---------------------------------

6.290.3.12 `virtual std::string decaf::io::DataInput::readString ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

#### Returns

string object containing the string read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.13 `virtual unsigned char decaf::io::DataInput::readUnsignedByte ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

#### Returns

the 8 bit unsigned value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.14 `virtual unsigned short decaf::io::DataInput::readUnsignedShort ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [pure virtual]

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) << 8) | (b \& 0xff)$

### Returns

the 16 bit unsigned short read.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.290.3.15 virtual std::string decaf::io::DataInput::readUTF ( ) throw  
( decaf::io::IOException, decaf::io::EOFException,  
decaf::io::UTFDataFormatException ) [pure virtual]

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1546) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

### Returns

The decoded string read from stream.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.
<b><i>UTFDataFormatException</i></b> (p. 3897)	if the bytes are not valid modified UTF-8 values.

6.290.3.16 virtual long long decaf::io::DataInput::skipBytes ( long long *num* ) throw (   
io::IOException ) [pure virtual]

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1789). The actual number of bytes skipped is returned.

**Parameters**

<i>num</i>	The number of bytes to skip over.
------------	-----------------------------------

**Returns**

the total number of bytes skipped.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataInput.h**

**6.291 decaf::io::DataInputStream Class Reference**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

```
#include <src/main/decaf/io/DataInputStream.h>
```

Inheritance diagram for decaf::io::DataInputStream:

**Public Member Functions**

- **DataInputStream** (**InputStream** \***inputStream**, bool **own**=false)  
*Creates a **DataInputStream** (p. 1532) that uses the specified underlying **InputStream** (p. 2002).*
- virtual ~**DataInputStream** ()
- virtual bool **readBoolean** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.*
- virtual char **readByte** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads and returns one input byte.*
- virtual unsigned char **readUnsignedByte** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.*
- virtual char **readChar** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an input char and returns the char value.*

- virtual double **readDouble** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a double value.*
- virtual float **readFloat** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns a float value.*
- virtual int **readInt** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads four input bytes and returns an int value.*
- virtual long long **readLong** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads eight input bytes and returns a long value.*
- virtual short **readShort** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns a short value.*
- virtual unsigned short **readUnsignedShort** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads two input bytes and returns an int value in the range 0 through 65535.*
- virtual std::string **readString** () throw ( decaf::io::IOException, decaf::io::EOFException )  
*Reads an NULL terminated ASCII string to the stream and returns the string to the caller.*
- virtual std::string **readLine** () throw ( decaf::io::IOException )  
*Reads the next line of text from the input stream.*
- virtual std::string **readUTF** () throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException )  
*Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.*
- virtual void **readFully** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads some bytes from an input stream and stores them into the buffer array buffer.*
- virtual void **readFully** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Reads length bytes from an input stream.*
- virtual long long **skipBytes** (long long num) throw ( io::IOException )  
*Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.*

### 6.291.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 2002) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

**DataInputStream** (p. 1532) `os = new DataInputStream (p. 1532)( new InputStream() (p. 2004), true )`

#### Since

1.0

### 6.291.2 Constructor & Destructor Documentation

6.291.2.1 `decaf::io::DataInputStream::DataInputStream ( InputStream * inputStream, bool own = false )`

Creates a **DataInputStream** (p. 1532) that uses the specified underlying **InputStream** (p. 2002).

#### Parameters

<i>inputStream</i>	the <b>InputStream</b> (p. 2002) instance to wrap.
<i>own</i>	indicates if this class owns the wrapped string defaults to false.

6.291.2.2 `virtual decaf::io::DataInputStream::~~DataInputStream ( ) [virtual]`

### 6.291.3 Member Function Documentation

6.291.3.1 `virtual bool decaf::io::DataInputStream::readBoolean ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]`

Reads in one byte and returns true if that byte is nonzero, false if that byte is zero.

#### Returns

the boolean value of the read in byte (0=false, 1=true).

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O Error occurs.
<b>EOFException</b> (p. 1789)	if the end of input is reached.



6.291.3.2 virtual char decaf::io::DataInputStream::readByte ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

#### Returns

the 8-bit value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.3 virtual char decaf::io::DataInputStream::readChar ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as `readByte`

#### Returns

the 8 bit char read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.4 virtual double decaf::io::DataInputStream::readDouble ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the `readlong` method, then converting this long value to a double in exactly the manner of the method `Double::longBitsToDouble`.

#### Returns

the double value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.5 `virtual float decaf::io::DataInputStream::readFloat ( ) throw ( decaf::io::IOException, decaf::io::EOFException )` [virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method `Float::intBitsToFloat`.

**Returns**

the float value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.6 `virtual void decaf::io::DataInputStream::readFully ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: \* length bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1789) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2103) other than **EOFException** (p. 1789) is thrown.

If buffer is NULL, a NullPointerException is thrown. If offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If length is zero, then no bytes are read. Otherwise, the first byte read is stored into element `buffer[offset]`, the next one into `buffer[offset+1]`, and so on. The number of bytes read is, at most, equal to length.

**Parameters**

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.
<i>offset</i>	The location in buffer to start writing.
<i>length</i>	The number of bytes to read from the buffer.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.
<i>NullPointerException</i>	if the buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size.

6.291.3.7 virtual void decaf::io::DataInputStream::readFully ( unsigned char \* *buffer*,  
int *size* ) throw ( decaf::io::IOException, decaf::io::EOFException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Reads some bytes from an input stream and stores them into the buffer array *buffer*.

The number of bytes read is equal to the length of *buffer*.

This method blocks until one of the following conditions occurs: \* *buffer*'s size bytes of input data are available, in which case a normal return is made. \* End of file is detected, in which case an **EOFException** (p. 1789) is thrown. \* An I/O error occurs, in which case an **IOException** (p. 2103) other than **EOFException** (p. 1789) is thrown.

If *buffer* size is zero, then no bytes are read. Otherwise, the first byte read is stored into element *b*[0], the next one into *buffer*[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of *buffer* have been updated with data from the input stream.

**Parameters**

<i>buffer</i>	The byte array to insert read data into.
<i>size</i>	The size in bytes of the given byte buffer.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.
<i>IndexOutOfBoundsException</i>	if the size value is negative.

6.291.3.8 virtual int decaf::io::DataInputStream::readInt ( ) throw ( decaf::io::IOException,  
decaf::io::EOFException ) [virtual]

Reads four input bytes and returns an int value.

Let *a* be the first byte read, *b* be the second byte, *c* be the third byte, and *d* be the fourth byte. The value returned is:

((*a* & 0xff) << 24) | ((*b* & 0xff) << 16) | ((*c* & 0xff) << 8) | (*d* & 0xff)

**Returns**

the int value read.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.9 `virtual std::string decaf::io::DataInputStream::readLine ( ) throw ( decaf::io::IOException ) [virtual]`

Reads the next line of text from the input stream.

It reads successive bytes, converting each byte to an ASCII char separately, until it encounters a line terminator or end of file; the characters read are then returned as a standard String. Note that because this method processes bytes, it does not support input of the full Unicode character set.

If end of file is encountered before even one byte can be read, then an empty string is returned. Otherwise, each byte that is read is converted to type char. If the character ' ' is encountered, it is discarded and reading ceases. If the character " is encountered, it is discarded and, if the following byte converts to the character ' ' , then that is discarded also; reading then ceases. If end of file is encountered before either of the characters ' ' and " is encountered, reading ceases. Once reading has ceased, a String is returned that contains all the characters read and not discarded, taken in order.

**Returns**

the next line of text read from the input stream or empty string if at EOF.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
--	-------------------------

6.291.3.10 `virtual long long decaf::io::DataInputStream::readLong ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]`

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is:

$((\text{long})(a \& 0\text{xff}) \ll 56) \mid ((\text{long})(b \& 0\text{xff}) \ll 48) \mid ((\text{long})(c \& 0\text{xff}) \ll 40) \mid ((\text{long})(d$

$\& 0xff) << 32) \mid ((\text{long})(e \& 0xff) << 24) \mid ((\text{long})(f \& 0xff) << 16) \mid ((\text{long})(g \& 0xff) << 8) \mid ((\text{long})(h \& 0xff)))$

### Returns

the 64 bit long long read.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.11 virtual short decaf::io::DataInputStream::readShort ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is:

$(\text{short})((a << 8) \mid (b \& 0xff))$

### Returns

the 16 bit short value read.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.12 virtual std::string decaf::io::DataInputStream::readString ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]

Reads an NULL terminated ASCII string to the stream and returns the string to the caller.

### Returns

string object containing the string read.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.13 `virtual unsigned char decaf::io::DataInputStream::readUnsignedByte ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]`

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

#### Returns

the 8 bit unsigned value read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.14 `virtual unsigned short decaf::io::DataInputStream::readUnsignedShort ( ) throw ( decaf::io::IOException, decaf::io::EOFException ) [virtual]`

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is:

$((a \& 0xff) \ll 8) | (b \& 0xff)$

#### Returns

the 16 bit unsigned short read.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.

6.291.3.15 `virtual std::string decaf::io::DataInputStream::readUTF ( ) throw ( decaf::io::IOException, decaf::io::EOFException, decaf::io::UTFDataFormatException ) [virtual]`

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1546) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

**Returns**

The decoded string read from stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
<b><i>EOFException</i></b> (p. 1789)	if the end of input is reached.
<b><i>UTFDataFormatException</i></b> (p. 3897)	if the bytes are not valid modified UTF-8 values.

6.291.3.16 `virtual long long decaf::io::DataInputStream::skipBytes ( long long num ) throw ( io::IOException ) [virtual]`

Makes an attempt to skip over *n* bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1789). The actual number of bytes skipped is returned.

**Parameters**

<i>num</i>	The number of bytes to skip over.
------------	-----------------------------------

**Returns**

the total number of bytes skipped.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O Error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInputStream.h`

**6.292 decaf::io::DataOutput Class Reference**

The **DataOutput** (p. 1541) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

```
#include <src/main/decaf/io/DataOutput.h>
```

## Public Member Functions

- virtual `~DataOutput ()`
- virtual void **writeBoolean** (bool value)=0 throw ( decaf::io::IOException )  
*Writes a boolean to the underlying output stream as a 1-byte value.*
- virtual void **writeByte** (unsigned char value)=0 throw ( decaf::io::IOException )  
*Writes out a byte to the underlying output stream as a 1-byte value.*
- virtual void **writeShort** (short value)=0 throw ( decaf::io::IOException )  
*Writes a short to the underlying output stream as two bytes, high byte first.*
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( decaf::io::IOException )  
*Writes a unsigned short to the bytes message stream as a 2 byte value.*
- virtual void **writeChar** (char value)=0 throw ( decaf::io::IOException )  
*Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.*
- virtual void **writeInt** (int value)=0 throw ( decaf::io::IOException )  
*Writes an int to the underlying output stream as four bytes, high byte first.*
- virtual void **writeLong** (long long value)=0 throw ( decaf::io::IOException )  
*Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.*
- virtual void **writeFloat** (float value)=0 throw ( decaf::io::IOException )  
*Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.*
- virtual void **writeDouble** (double value)=0 throw ( decaf::io::IOException )  
*Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.*
- virtual void **writeBytes** (const std::string &value)=0 throw ( decaf::io::IOException )  
*Writes out the string to the underlying output stream as a sequence of bytes.*
- virtual void **writeChars** (const std::string &value)=0 throw ( decaf::io::IOException )  
*Writes a string to the underlying output stream as a sequence of characters.*
- virtual void **writeUTF** (const std::string &value)=0 throw ( decaf::io::IOException, decaf::io::UTFDataFormatException )  
*Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.*

### 6.292.1 Detailed Description

The **DataOutput** (p. 1541) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.

There is also a facility for converting Strings into the Java standard modified UTF-8 format and writing the resulting series of bytes.

If a method in this interface encounters an error while writing it will throw an **IOException** (p. 2103).



**See also**

**DataInput** (p. 1523)  
**DataOutputStream** (p. 1546)

**Since**

1.0

**6.292.2 Constructor & Destructor Documentation**

6.292.2.1 virtual decaf::io::DataOutput::~DataOutput( ) [inline, virtual]

**6.292.3 Member Function Documentation**

6.292.3.1 virtual void decaf::io::DataOutput::writeBoolean( bool *value* ) throw ( decaf::io::IOException ) [pure virtual]

Writes a boolean to the underlying output stream as a 1-byte value.

The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

**Parameters**

<i>value</i>	The boolean to write as a byte (1=true, 0=false).
--------------	---

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.2 virtual void decaf::io::DataOutput::writeByte( unsigned char *value* ) throw ( decaf::io::IOException ) [pure virtual]

Writes out a byte to the underlying output stream as a 1-byte value.

If no exception is thrown, the counter written is incremented by 1.

**Parameters**

<i>value</i>	The unsigned char value to write.
--------------	-----------------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.3 `virtual void decaf::io::DataOutput::writeBytes ( const std::string & value ) throw ( decaf::io::IOException ) [pure virtual]`

Writes out the string to the underlying output stream as a sequence of bytes.

Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

#### Parameters

<i>value</i>	The vector of bytes to write.
--------------	-------------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.4 `virtual void decaf::io::DataOutput::writeChar ( char value ) throw ( decaf::io::IOException ) [pure virtual]`

Writes out a char to the underlying output stream as a one byte value. If no exception is thrown, the counter written is incremented by 1.

#### Parameters

<i>value</i>	The signed char value to write.
--------------	---------------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.5 `virtual void decaf::io::DataOutput::writeChars ( const std::string & value ) throw ( decaf::io::IOException ) [pure virtual]`

Writes a string to the underlying output stream as a sequence of characters.

Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

#### Parameters

<i>value</i>	The string value to write as raw bytes.
--------------	---

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.6 `virtual void decaf::io::DataOutput::writeDouble ( double value ) throw ( decaf::io::IOException )` [pure virtual]

Converts the double argument to a long using the `doubleToLongBits` method in class `Double`, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 8.

#### Parameters

<i>value</i>	The 64bit double value to write.
--------------	----------------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.7 `virtual void decaf::io::DataOutput::writeFloat ( float value ) throw ( decaf::io::IOException )` [pure virtual]

Converts the float argument to an int using the `floatToIntBits` method in class `Float`, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 4.

#### Parameters

<i>value</i>	The 32bit floating point value to write.
--------------	--

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.8 `virtual void decaf::io::DataOutput::writeInt ( int value ) throw ( decaf::io::IOException )` [pure virtual]

Writes an int to the underlying output stream as four bytes, high byte first.

If no exception is thrown, the counter written is incremented by 4.

#### Parameters

<i>value</i>	The signed integer value to write.
--------------	------------------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.9 `virtual void decaf::io::DataOutput::writeLong ( long long value ) throw ( decaf::io::IOException )` [pure virtual]

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

If no exception is thrown, the counter written is incremented by 8.

#### Parameters

<i>value</i>	The signed 64bit long value to write.
--------------	---------------------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.10 `virtual void decaf::io::DataOutput::writeShort ( short value ) throw ( decaf::io::IOException )` [pure virtual]

Writes a short to the underlying output stream as two bytes, high byte first.

If no exception is thrown, the counter written is incremented by 2.

#### Parameters

<i>value</i>	The signed short value to write.
--------------	----------------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.11 `virtual void decaf::io::DataOutput::writeUnsignedShort ( unsigned short value ) throw ( decaf::io::IOException )` [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

#### Parameters

<i>value</i>	The unsigned short to write to the stream.
--------------	--

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error is encountered.
--	---------------------------------

6.292.3.12 virtual void decaf::io::DataOutput::writeUTF ( const std::string & *value* ) throw ( decaf::io::IOException, decaf::io::UTFDataFormatException ) [pure virtual]

Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

The first two bytes written are indicate its encoded length followed by the rest of the string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

#### Parameters

<i>value</i>	The string value value to write as modified UTF-8.
--------------	--

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error is encountered.
<b>UTFDataFormatException</b> (p. 3897)	if the encoded size if greater than 65535

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutput.h**

## 6.293 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

```
#include <src/main/decaf/io/DataOutputStream.h>
```

Inheritance diagram for decaf::io::DataOutputStream:

#### Public Member Functions

- **DataOutputStream** (**OutputStream** \***outputStream**, bool **own**=false)  
*Creates a new data output stream to write data to the specified underlying output stream.*
- virtual ~**DataOutputStream** ()
- virtual long long **size** () const  
*Returns the current value of the counter written, the number of bytes written to this data output stream so far.*
- virtual void **writeBoolean** (bool **value**) throw ( **IOException** )

- virtual void **writeByte** (unsigned char value) throw ( IOException )
- virtual void **writeShort** (short value) throw ( IOException )
- virtual void **writeUnsignedShort** (unsigned short value) throw ( IOException )
- virtual void **writeChar** (char value) throw ( IOException )
- virtual void **writeInt** (int value) throw ( IOException )
- virtual void **writeLong** (long long value) throw ( IOException )
- virtual void **writeFloat** (float value) throw ( IOException )
- virtual void **writeDouble** (double value) throw ( IOException )
- virtual void **writeBytes** (const std::string &value) throw ( IOException )
- virtual void **writeChars** (const std::string &value) throw ( IOException )
- virtual void **writeUTF** (const std::string &value) throw ( IOException, UTFDataFormatException )

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \***buffer**, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### Protected Attributes

- long long **written**
- unsigned char **buffer** [8]

#### 6.293.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

An application can then use a data input stream to read the data back in.

## 6.293.2 Constructor & Destructor Documentation

6.293.2.1 `decaf::io::DataOutputStream::DataOutputStream ( OutputStream * outputStream, bool own = false )`

Creates a new data output stream to write data to the specified underlying output stream.

### Parameters

<i>output-Stream</i>	a stream to wrap with this one.
<i>own</i>	true if this objects owns the stream that it wraps.

6.293.2.2 `virtual decaf::io::DataOutputStream::~~DataOutputStream ( ) [virtual]`

## 6.293.3 Member Function Documentation

6.293.3.1 `virtual void decaf::io::DataOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.293.3.2 `virtual void decaf::io::DataOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException ) [protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.293.3.3 `virtual long long decaf::io::DataOutputStream::size ( ) const [inline, virtual]`

Returns the current value of the counter written, the number of bytes written to this data output stream so far.

If the counter overflows, it will be wrapped to **decaf::lang::Long::MAX\_VALUE** (p. 2392).

### Returns

the value of the written field.

6.293.3.4 `virtual void decaf::io::DataOutputStream::writeBoolean ( bool value ) throw ( IOException ) [virtual]`

- 6.293.3.5 `virtual void decaf::io::DataOutputStream::writeByte ( unsigned char value ) throw ( IOException )` [virtual]
- 6.293.3.6 `virtual void decaf::io::DataOutputStream::writeBytes ( const std::string & value ) throw ( IOException )` [virtual]
- 6.293.3.7 `virtual void decaf::io::DataOutputStream::writeChar ( char value ) throw ( IOException )` [virtual]
- 6.293.3.8 `virtual void decaf::io::DataOutputStream::writeChars ( const std::string & value ) throw ( IOException )` [virtual]
- 6.293.3.9 `virtual void decaf::io::DataOutputStream::writeDouble ( double value ) throw ( IOException )` [virtual]
- 6.293.3.10 `virtual void decaf::io::DataOutputStream::writeFloat ( float value ) throw ( IOException )` [virtual]
- 6.293.3.11 `virtual void decaf::io::DataOutputStream::writeInt ( int value ) throw ( IOException )` [virtual]
- 6.293.3.12 `virtual void decaf::io::DataOutputStream::writeLong ( long long value ) throw ( IOException )` [virtual]
- 6.293.3.13 `virtual void decaf::io::DataOutputStream::writeShort ( short value ) throw ( IOException )` [virtual]
- 6.293.3.14 `virtual void decaf::io::DataOutputStream::writeUnsignedShort ( unsigned short value ) throw ( IOException )` [virtual]
- 6.293.3.15 `virtual void decaf::io::DataOutputStream::writeUTF ( const std::string & value ) throw ( IOException, UTFDataFormatException )` [virtual]

#### 6.293.4 Field Documentation

- 6.293.4.1 `unsigned char decaf::io::DataOutputStream::buffer[8]` [protected]
- 6.293.4.2 `long long decaf::io::DataOutputStream::written` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataOutputStream.h`

## 6.294 activemq::commands::DataResponse Class Reference

```
#include <src/main/activemq/commands/DataResponse.h>
```



Inheritance diagram for activemq::commands::DataResponse:

## Public Member Functions

- **DataResponse** ()
- virtual **~DataResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataResponse \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getData** () const
- virtual **Pointer**< **DataStructure** > & **getData** ()
- virtual void **setData** (const **Pointer**< **DataStructure** > &data)

## Static Public Attributes

- static const unsigned char **ID\_DATARESPONSE** = 32

## Protected Attributes

- **Pointer**< **DataStructure** > data

## 6.294.1 Constructor & Destructor Documentation

6.294.1.1 **activemq::commands::DataResponse::DataResponse** ( )

6.294.1.2 **virtual activemq::commands::DataResponse::~~DataResponse** ( ) *[virtual]*

## 6.294.2 Member Function Documentation

6.294.2.1 **virtual `DataResponse*` `activemq::commands::DataResponse::cloneDataStructure ( ) const`** `[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **`activemq::commands::Response`** (p. 3228).

6.294.2.2 **virtual void `activemq::commands::DataResponse::copyDataStructure ( const DataStructure * src )`** `[virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **`activemq::commands::Response`** (p. 3229).

6.294.2.3 **virtual bool `activemq::commands::DataResponse::equals ( const DataStructure * value ) const`** `[virtual]`

Compares the **`DataStructure`** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if `DataStructure`'s are Equal.

Reimplemented from **`activemq::commands::Response`** (p. 3229).

6.294.2.4 **virtual `Pointer<DataStructure>&` `activemq::commands::DataResponse::getData ( )`** `[virtual]`

6.294.2.5 **virtual const `Pointer<DataStructure>&` `activemq::commands::DataResponse::getData ( ) const`** `[virtual]`

## 6.295 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller

### Class Reference

---

1557

6.294.2.6 `virtual unsigned char activemq::commands::DataResponse::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaller share.

#### Returns

new **DataSet** (p. 1628) type copy.

Reimplemented from **activemq::commands::Response** (p. 3230).

6.294.2.7 `virtual void activemq::commands::DataResponse::setData ( const Pointer<DataSet> & data ) [virtual]`

6.294.2.8 `virtual std::string activemq::commands::DataResponse::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3230).

### 6.294.3 Field Documentation

6.294.3.1 `Pointer<DataSet> activemq::commands::DataResponse::data [protected]`

6.294.3.2 `const unsigned char activemq::commands::DataResponse::ID_DATARESPONSE = 32 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataResponse.h**

## 6.295 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1553).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**:

## Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.295.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1553).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.295.2 Constructor & Destructor Documentation

6.295.2.1 **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::DataResponseMarshaller**  
 ( ) [inline]

6.295.2.2 **virtual activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::~~DataResponseMarshaller**  
 ( ) [inline, virtual]

### 6.295.3 Member Function Documentation

6.295.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

6.295.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

6.295.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

```

6.295.3.4 virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

```

6.295.3.5 virtual int activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

6.295.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
 ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**  
 (p. 3249).

6.295.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**  
 (p. 3250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DataResponseMarshaller.h**

## 6.296 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1557).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.296.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1557).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.296.2 Constructor & Destructor Documentation

6.296.2.1 `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]`

6.296.2.2 `virtual activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::~~DataResponseMarshaller ( ) [inline, virtual]`

## 6.296.3 Member Function Documentation

6.296.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3261).

6.296.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3261).

6.296.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

```
6.296.3.4  virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262).

```
6.296.3.5  virtual int activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.296 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller

### Class Reference 1565

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262).

```
6.296.3.6 virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3263).

```
6.296.3.7 virtual void activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3264).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DataResponseMarshaller.h**

## 6.297 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1561).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.297.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1561).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.297.2 Constructor & Destructor Documentation

6.297.2.1 `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]`

6.297.2.2 `virtual activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::~~DataResponseMarshaller ( ) [inline, virtual]`

## 6.297.3 Member Function Documentation

6.297.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3242).

6.297.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3242).

6.297.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.297.3.4  virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.297.3.5  virtual int activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.297 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller

### Class Reference 1569

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

6.297.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

6.297.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataResponseMarshaller.h**

## 6.298 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1565).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller:

#### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.298.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1565).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.298.2 Constructor & Destructor Documentation

6.298.2.1    `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]`

6.298.2.2    `virtual activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::~~DataResponseMarshaller ( ) [inline, virtual]`

## 6.298.3 Member Function Documentation

6.298.3.1    `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3251).

6.298.3.2    `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

6.298.3.3    `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

```
6.298.3.4  virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253).

```
6.298.3.5  virtual int activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.298 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller

### Class Reference 1573

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253).

6.298.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254).

6.298.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h**

## 6.299 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1569).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.299.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1569).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.299.2 Constructor & Destructor Documentation

6.299.2.1 `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]`

6.299.2.2 `virtual activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::~~DataResponseMarshaller ( ) [inline, virtual]`

## 6.299.3 Member Function Documentation

6.299.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3237).

6.299.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

6.299.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

```
6.299.3.4  virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

```
6.299.3.5  virtual int activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.299 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller

### Class Reference 1577

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

6.299.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

6.299.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DataResponseMarshaller.h**

## 6.300 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1573).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller:

### Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.300.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1573).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.300.2 Constructor & Destructor Documentation

6.300.2.1 `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::DataResponseMarshaller ( ) [inline]`

6.300.2.2 `virtual activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::~~DataResponseMarshaller ( ) [inline, virtual]`

## 6.300.3 Member Function Documentation

6.300.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

6.300.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

6.300.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.300.3.4  virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.300.3.5  virtual int activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.300 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller

### Class Reference 1581

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

6.300.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

6.300.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3259).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataResponseMarshaller.h**

## 6.301 activemq::wireformat::openwire::marshal::DataStreamMarshaller

### Class Reference

Base class for all classes that marshal commands for Openwire.

```
#include <src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::DataStreamMarshaller:

#### Public Member Functions

- virtual **~DataStreamMarshaller** ()
- virtual unsigned char **getDataStructureType** () const =0  
*Gets the DataStructureType that this class marshals/unmarshals.*
- virtual **commands::DataStructure \* createObject** () const =0  
*Creates a new instance of the class that this class is a marshaling director for.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **utils::BooleanStream** \*bs)=0 throw ( **decaf::io::IOException** )  
*Tight Marhsal to the given stream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataOutputStream** \*ds, **utils::BooleanStream** \*bs)=0 throw ( **decaf::io::IOException** )  
*Tight Marhsal to the given stream.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataInputStream** \*dis, **utils::BooleanStream** \*bs)=0 throw ( **decaf::io::IOException** )  
*Tight Un-marhsal to the given stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataOutputStream** \*ds)=0 throw ( **decaf::io::IOException** )  
*Tight Marhsal to the given stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*format, **commands::DataStructure** \*command, **decaf::io::DataInputStream** \*dis)=0 throw ( **decaf::io::IOException** )  
*Loose Un-marhsal to the given stream.*

#### 6.301.1 Detailed Description

Base class for all classes that marshal commands for Openwire.

## 6.301.2 Constructor & Destructor Documentation

6.301.2.1 virtual activemq::wireformat::openwire::marshal::DataStreamMarshaller::~~DataStreamMarshaller  
( ) [inline, virtual]

## 6.301.3 Member Function Documentation

6.301.3.1 virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject ( )  
const [pure virtual]

Creates a new instance of the class that this class is a marshaling director for.

### Returns

newly allocated Command

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 225), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 349), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 376), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 422), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 465), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 528), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 583), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 616), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 645), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 673), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 841), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 872), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1251), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1283), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 1314), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1344), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1387), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1415), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1448), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1476), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1509), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1574), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1709), **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** (p. 1742), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1826), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1920), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2074), **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** (p. 2140), **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller** (p. 2169), **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** (p. 2191), **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** (p. 2223), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2250),

**activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2284),  
**activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2331),  
**activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2543), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2583),  
**activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 2613), **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller** (p. 2649),  
**activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 2717), **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller** (p. 2770),  
**activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2893),  
**activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 3009),  
**activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller** (p. 3040), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 3057), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 3154), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 3171),  
**activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 3202),  
**activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256), **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller** (p. 3345), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 3425), **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller** (p. 3625), **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3794), **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller** (p. 3940), **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3977), **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 191), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 241), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 361), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 434), **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 477), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 540), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 595), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 624), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 657), **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 685), **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller** (p. 853), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 884), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1263), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1271), **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller** (p. 1302), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1332), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1375), **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller** (p. 1403), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1436), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1464), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1497), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1562), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1697), **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller** (p. 1730), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1810), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1908), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2062),

activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2124),  
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2153),  
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2175),  
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2207),  
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2234),  
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 2272),  
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 2315),  
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2531),  
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2567),  
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
(p. 2600), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2629),  
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2701),  
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2750),  
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2875),  
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2989),  
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3020),  
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3053),  
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3142),  
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 3179),  
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 3206),  
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3242),  
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3325), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller  
(p. 3369), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller  
(p. 3421), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller  
(p. 3641), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller  
(p. 3810), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller  
(p. 3932), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
(p. 3969), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
(p. 179), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
(p. 221), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
(p. 345), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
(p. 372), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
(p. 418), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
(p. 461), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
(p. 524), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
(p. 579), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
(p. 608), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
(p. 637), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
(p. 665), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 833),  
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 864),  
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1243),  
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1275),  
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1306),  
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1336),  
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1379),  
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1407),  
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1440),  
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1468),  
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1501),  
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1566),

**activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1701),  
**activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller** (p. 1734),  
**activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1814),  
**activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1912),  
**activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2066),  
**activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** (p. 2132),  
**activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller** (p. 2157),  
**activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** (p. 2179),  
**activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** (p. 2211),  
**activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 2238),  
**activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2268),  
**activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2319),  
**activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 2535), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 2571),  
**activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 2604), **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller** (p. 2641),  
**activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 2709), **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller** (p. 2762),  
**activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2884),  
**activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2997),  
**activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller** (p. 3028), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 3065), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 3150), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 3175),  
**activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 3210),  
**activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3251), **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** (p. 3341), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 3365), **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller** (p. 3433),  
**activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3621), **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 3798),  
**activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3944), **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3981),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 229),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 353), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 380),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 426), **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller** (p. 469),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 532), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 587),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 612), **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 641),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller** (p. 669), **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller** (p. 837),  
**activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** (p. 868), **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller** (p. 1247),  
**activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller** (p. 1279), **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller** (p. 1310),  
**activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller** (p. 1340), **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller** (p. 1383),



activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1411), activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1444), activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1472), activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1505), activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1570), activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1705), activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1738), activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1822), activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1916), activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 2070), activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 2136), activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 2165), activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2187), activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2219), activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2242), activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 2280), activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2327), activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2539), activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2579), activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (p. 2608), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2633), activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2713), activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2766), activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2888), activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2993), activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3024), activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3049), activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3162), activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 3191), activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 3198), activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3237), activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3329), activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3437), activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3633), activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3806), activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3936), activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3973), activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 195), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 233), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 357), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 384), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 430), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 473), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (p. 536), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (p. 591), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (p. 620), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (p. 649), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (p. 677), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 845), activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 876), ac-

tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1255),  
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1287),  
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1318),  
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1348),  
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1391),  
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1419), ac-  
tivemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1452), ac-  
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1480),  
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1513),  
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1554),  
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1717),  
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1746),  
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1818),  
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1924),  
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2078),  
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2128),  
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2149),  
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2195),  
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2215),  
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2246),  
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (p. 2276),  
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (p. 2323),  
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2547), ac-  
tivemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2575),  
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
(p. 2617), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2637),  
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2705), ac-  
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2758),  
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2880),  
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3001),  
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3032), ac-  
tivemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3061), ac-  
tivemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3158), ac-  
tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (p. 3187),  
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (p. 3218),  
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3247), ac-  
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3337), activemq::wireformat::openwi-  
(p. 3357), activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller  
(p. 3429), activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller  
(p. 3629), activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller  
(p. 3790), activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller  
(p. 3924), activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller  
(p. 3985), activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller  
(p. 199), activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
(p. 237), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
(p. 365), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
(p. 392), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
(p. 438), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
(p. 481), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
(p. 544), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller

(p. 599), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 628), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 653), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 681), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 849), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 880), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1259), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1291), `activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1322), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1352), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1395), `activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1423), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1456), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1484), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1517), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1558), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1713), `activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1726), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1806), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1904), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2058), `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2120), `activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2161), `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2183), `activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2203), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2230), `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2264), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2311), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2527), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2587), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2596), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2645), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2721), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2754), `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2871), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3005), `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3036), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3069), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3146), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3183), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3214), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3261), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3333), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3417), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3637), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3802), `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3928), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3965).

```
6.301.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::DataStreamMarshaller::getDataStructureType
( ) const [pure virtual]
```

Gets the DataStructureType that this class marshals/unmarshals.

### Returns

byte Id of this classes DataStructureType

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 183), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 226), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 376), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 422), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 466), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 529), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 584), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 616), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 645), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 673), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 841), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 872), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1251), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1283), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1314), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1344), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1387), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1415), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1448), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1476), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1510), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1575), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1709), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1742), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1826), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1921), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2074), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2141), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2169), `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2192), `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2223), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2250), `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2284), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2332), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2544), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2584), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2613), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2649), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2717), `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2771),

activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2893),  
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3009),  
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3040), ac-  
tivismq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3057), ac-  
tivismq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3155), ac-  
tivismq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3171),  
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3202),  
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3256), ac-  
tivismq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3346), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 3361), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 3425), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3626), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3795), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3940), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 3978), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 191), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 242), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 362), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 388), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 434), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 478), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 541), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 596), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 624), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 657), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 685), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 853), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 884), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1263), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1271), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1302), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1332), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1375), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1403), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1436), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1464), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1497), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1562), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1697), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1730), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1810), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1909), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2062), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2125), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2153), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2176), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2207), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2234), activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 2272), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 2316), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2532), ac-

tivemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2568),  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
 (p. 2600), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2629),  
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2701), ac-  
 tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2751),  
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2876),  
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2989),  
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3020), ac-  
 tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3053), ac-  
 tivemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3143), ac-  
 tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 3179),  
 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 3206),  
 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3242), ac-  
 tivemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3326), activemq::wireformat::openwi-  
 re (p. 3369), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller  
 (p. 3421), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller  
 (p. 3642), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller  
 (p. 3811), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller  
 (p. 3932), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller  
 (p. 3970), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller  
 (p. 179), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
 (p. 222), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
 (p. 346), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
 (p. 372), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
 (p. 418), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
 (p. 462), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
 (p. 524), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
 (p. 580), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
 (p. 608), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
 (p. 637), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
 (p. 665), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 833),  
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 864), ac-  
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1243),  
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1275),  
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1306),  
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1336),  
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1379),  
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1407), ac-  
 tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1440), ac-  
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1468),  
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1501),  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1567),  
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1701),  
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1734),  
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1814),  
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1913),  
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2066),  
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2133),  
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2157),  
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2180),

activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2211),  
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2238),  
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 2268),  
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 2320),  
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2536), ac-  
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2572),  
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
(p. 2605), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2641),  
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2709), ac-  
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2763),  
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2884),  
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2997),  
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3028), ac-  
tivemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3065), ac-  
tivemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3151), ac-  
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 3175),  
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 3210),  
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3252), ac-  
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3342), activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller  
(p. 3365), activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller  
(p. 3433), activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller  
(p. 3622), activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller  
(p. 3799), activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller  
(p. 3944), activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller  
(p. 187), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
(p. 230), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
(p. 354), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
(p. 380), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
(p. 426), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
(p. 470), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
(p. 533), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
(p. 588), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
(p. 612), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
(p. 641), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
(p. 669), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 837),  
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 868), ac-  
tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1247),  
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1279),  
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1310),  
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1340),  
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1383),  
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1411), ac-  
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1444), ac-  
tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1472),  
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1506),  
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1571),  
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1705),  
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1738),  
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1822),

`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1917),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2070),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2137),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2165),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2188),  
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2219),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2242),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2280),  
`activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2328),  
`activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2540), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2580),  
`activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2609), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2633),  
`activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2713), `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2767),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2889),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2993),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3024), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3049), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3163), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3191),  
`activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3198),  
`activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3238), `activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3330), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3373), `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3437),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3634), `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3807),  
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3936), `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 3974),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 195), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 234),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 384),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 430), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 474),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 537), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 592),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 620), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 649),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 677), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 845),  
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 876), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1255),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1287), `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1318),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1348), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1391),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1419), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1452), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1480),



activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1514),  
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1554),  
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1717),  
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1746),  
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1818),  
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1925),  
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2078),  
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2129),  
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2149),  
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2196),  
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2215),  
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2246),  
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (p. 2276),  
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (p. 2324),  
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2548),  
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2576),  
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (p. 2617),  
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2637),  
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2705),  
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2759),  
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2880),  
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3001),  
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3032),  
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3061),  
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3159),  
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (p. 3187),  
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (p. 3218),  
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3247),  
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3338),  
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 3357),  
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 3429),  
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3630),  
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3790),  
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3924),  
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3986),  
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller (p. 199),  
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller (p. 238),  
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller (p. 366),  
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller (p. 392),  
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller (p. 438),  
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller (p. 482),  
activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller (p. 545),  
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller (p. 600),  
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller (p. 628),  
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller (p. 653),  
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller (p. 681),  
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 849),  
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 880),  
activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1259),  
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1291),  
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1322),

`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1352),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1395),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1423), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1456), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1484),  
`activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1518),  
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1558),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1713),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1726),  
`activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1806),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1905),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2058),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2121),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2161),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2184),  
`activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2203),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2230),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2264),  
`activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2312),  
`activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2527), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2588),  
`activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2596), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2645),  
`activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2721), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2755),  
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2871),  
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3005),  
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3036), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3069), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3147), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3183),  
`activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3214),  
`activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3261), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3334), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3417), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3638), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3803), `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3928), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3966).

```

6.301.3.3  virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal
( OpenWireFormat * format, commands::DataStructure * command,
  decaf::io::DataOutputStream * ds ) throw ( decaf::io::IOException )
[pure virtual]
  
```

Tight Marshal to the given stream.

## Parameters

<i>format</i>	- The OpenwireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- DataOutputStream to marshal to

## Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 226), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 309), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 350), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 422), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 466), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 529), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 584), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 616), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 645), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 674), **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 842), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 872), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 1252), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 1283), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 1314), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1344), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1387), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1416), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1448), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1476), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1510), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1575), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1710), **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** (p. 1743), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1826), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1921), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2074), **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** (p. 2141), **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller** (p. 2169), **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** (p. 2192), **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** (p. 2223), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 2250), **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2284), **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2332), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 2544), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 2584), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

(p. 2613), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2650), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2671), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2717), `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2771), `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2893), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3009), `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3040), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3057), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3155), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3171), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3203), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3257), `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3346), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3361), `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3426), `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3767), `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3795), `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 3940), `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3978), `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 191), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 242), `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 321), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 434), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 478), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 541), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 567), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 596), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 624), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 657), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 686), `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 765), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 854), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 884), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1271), `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1302), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1332), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1375), `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1404), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1436), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1464), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1563), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1698), `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1731), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1810), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1909), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2062),

activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2125),  
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2153),  
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2176),  
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2207),  
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2234),  
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 2272),  
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 2316),  
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2532),  
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2568),  
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller  
(p. 2601), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2630),  
activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2663),  
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2701),  
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2751),  
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2876),  
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2989),  
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3020),  
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3053),  
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3143),  
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 3179),  
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 3207),  
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3243),  
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3326),  
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3369),  
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3422),  
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3771),  
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3811),  
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3932),  
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3970),  
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 179),  
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
(p. 222), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller  
(p. 305), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
(p. 346), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
(p. 373), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
(p. 418), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
(p. 462), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
(p. 525), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller  
(p. 552), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
(p. 580), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
(p. 608), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
(p. 637), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
(p. 666), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller  
(p. 731), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 834),  
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 864),  
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1244),  
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1275),  
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1306),  
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1336),  
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1379),

`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1440), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1468), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1567), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1702), `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1735), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1814), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1913), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2066), `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2133), `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2157), `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2180), `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2211), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2238), `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 2268), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2320), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2536), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2572), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2605), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2642), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2658), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2709), `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2763), `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2885), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2997), `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3028), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3065), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3151), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3175), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3252), `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3342), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3434), `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3622), `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3775), `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3799), `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 3944), `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3982), `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 187), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 313), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 381), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 426), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 470), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 560), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 588), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`

(p. 612), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
(p. 641), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`  
(p. 670), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`  
(p. 738), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 838),  
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 868), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1248),  
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1279),  
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1310),  
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1340),  
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1383),  
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1412), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1444), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1472),  
`activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1506),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1571),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1706),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1739),  
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1822),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1917),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2070),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2137),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2165),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2188),  
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2219),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2242),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2280),  
`activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2328),  
`activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2540), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2580),  
`activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
(p. 2609), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2634),  
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2667), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2713), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2767),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2889),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2993),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3024), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3049), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3163), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3191),  
`activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3199),  
`activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3238), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3330), `activemq::wireformat::openwire::marshal::`  
(p. 3373), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller`  
(p. 3438), `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller`  
(p. 3634), `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3779),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3807),  
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3936),  
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3974),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 195),

`activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 234), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 317), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 430), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 474), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 537), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 563), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 592), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 620), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 649), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 678), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 751), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 846), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 876), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1256), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1287), `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1318), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1348), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1391), `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1420), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1452), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1480), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1514), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1554), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1718), `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1747), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1818), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1925), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2078), `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2129), `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2149), `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2196), `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2215), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2246), `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2276), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2324), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2548), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2576), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2617), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2638), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2654), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2705), `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2759), `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2880), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3001), `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3032), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3061), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3159), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3187),



**activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 3219),  
**activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247), **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller** (p. 3338), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 3357), **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller** (p. 3430), **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3764), **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3791), **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller** (p. 3924), **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 3986), **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 199), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 238), **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller** (p. 325), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 366), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 438), **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller** (p. 482), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 545), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller** (p. 571), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller** (p. 600), **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller** (p. 628), **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 653), **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller** (p. 682), **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758), **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller** (p. 850), **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller** (p. 880), **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller** (p. 1260), **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller** (p. 1291), **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller** (p. 1322), **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller** (p. 1352), **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller** (p. 1395), **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller** (p. 1424), **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller** (p. 1456), **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller** (p. 1484), **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1518), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller** (p. 1714), **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller** (p. 1727), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1806), **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller** (p. 1905), **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2058), **activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller** (p. 2121), **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller** (p. 2161), **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller** (p. 2184), **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller** (p. 2203), **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller** (p. 2230), **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2264), **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2312), **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller** (p. 2528), **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller** (p. 2588), **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

(p. 2597), **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller** (p. 2646), **activemq::wireformat::openwire::marshal::v6::MessageMarshaller** (p. 2676), **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller** (p. 2721), **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller** (p. 2755), **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2872), **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller** (p. 3005), **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller** (p. 3036), **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller** (p. 3069), **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller** (p. 3147), **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller** (p. 3183), **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller** (p. 3215), **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261), **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller** (p. 3334), **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller** (p. 3353), **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller** (p. 3418), **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3638), **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller** (p. 3782), **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller** (p. 3803), **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller** (p. 3928), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3966).

```
6.301.3.4  virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal
( OpenWireFormat * format, commands::DataStructure * command,
  decaf::io::DataInputStream * dis ) throw ( decaf::io::IOException )
[pure virtual]
```

Loose Un-marhsal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 226), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 310), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 350), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 466), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 529), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 556), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 584), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 617), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 646), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**

(p. 674), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 746), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 842), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 873), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1252), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1284), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1345), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1388), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1449), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1477), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1510), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1575), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1710), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1743), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1827), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1921), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2075), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2141), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2170), `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2192), `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2224), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2251), `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2285), `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2332), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2544), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2584), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2614), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2650), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2672), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2718), `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2771), `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2894), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3010), `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3041), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3058), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3155), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3172), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3203), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3257), `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3346), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3362), `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3426), `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3626), `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3768), `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3795), `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 3941), `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3978), `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 242), `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`

(p. 322), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`  
(p. 362), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`  
(p. 389), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`  
(p. 435), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`  
(p. 478), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`  
(p. 541), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller`  
(p. 568), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`  
(p. 596), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`  
(p. 625), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`  
(p. 658), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`  
(p. 686), `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`  
(p. 766), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 854),  
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 885), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1264),  
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1272),  
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1303),  
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1333),  
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1376),  
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1404), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1437), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1465),  
`activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1498),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1563),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1698),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1731),  
`activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1811),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1909),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2063),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2125),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2154),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2176),  
`activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2208),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2235),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2273),  
`activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2316),  
`activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2532), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2568),  
`activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`  
(p. 2601), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2630),  
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2663), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2702), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2751),  
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2876),  
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2990),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3021), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3054), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3143), `ac-`  
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3180),  
`activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3207),  
`activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3243), `ac-`

tivemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3326), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3370), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3422), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3642), activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3772), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3811), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3933), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3970), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 180), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 222), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (p. 306), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 346), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 373), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 419), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 462), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 525), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (p. 552), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 580), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (p. 609), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 638), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (p. 666), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (p. 732), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 834), activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 865), activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1244), activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1276), activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1307), activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1337), activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1380), activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1408), activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1441), activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1469), activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1502), activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1567), activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1702), activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1735), activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1815), activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1913), activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2067), activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2133), activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2158), activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2180), activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2212), activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2239), activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 2269), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 2320), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2536), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2572), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2605), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2642), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2659), ac-

`tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2710), `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2763), `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2885), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2998), `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3029), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3066), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3151), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3176), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3253), `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3342), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3434), `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3622), `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3775), `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3799), `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 3945), `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3982), `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 314), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 381), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 427), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 470), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 533), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 560), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 588), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 613), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 642), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 670), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 739), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 838), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 869), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1248), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1311), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1341), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1384), `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1412), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1445), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1473), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1506), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1571), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1706), `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1739), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1823), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1917), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2071), `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2137), `activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2166),

activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 2188),  
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 2220),  
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 2243),  
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (p. 2281),  
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (p. 2328),  
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (p. 2540),  
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (p. 2580),  
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller  
(p. 2609), activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2634),  
activemq::wireformat::openwire::marshal::v4::MessageMarshaller (p. 2668),  
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2714),  
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2767),  
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2889),  
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2994),  
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 3025),  
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 3050),  
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 3163),  
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (p. 3192),  
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (p. 3199),  
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 3239),  
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3330),  
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3374),  
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3438),  
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3779),  
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3807),  
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3937),  
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3974),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 196),  
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller  
(p. 234), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller  
(p. 318), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller  
(p. 358), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller  
(p. 385), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller  
(p. 431), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller  
(p. 474), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller  
(p. 537), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller  
(p. 564), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller  
(p. 592), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller  
(p. 621), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller  
(p. 650), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller  
(p. 678), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller  
(p. 752), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 846),  
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 877),  
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1256),  
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1288),  
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1319),  
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1349),  
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1392),  
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1420),  
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1453), ac-

`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1481),  
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1514),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1555),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1718),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1747),  
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1819),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1925),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2079),  
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2129),  
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2150),  
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2196),  
`activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2216),  
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2247),  
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2277),  
`activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2324),  
`activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2548), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2576),  
`activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2618), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2638),  
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2655), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2706), `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2759),  
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2881),  
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3002),  
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3033), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3062), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3159), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3188),  
`activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3219),  
`activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3248), `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3338), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3430), `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3630), `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3764),  
`activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3791),  
`activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3925),  
`activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3986),  
`activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 200),  
`activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 238), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 326), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 439), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 482), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 545), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 572), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 600), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 629), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 654), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`



(p. 682), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 759), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 850), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 881), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1260), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1292), `activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1323), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1353), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1396), `activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1424), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1457), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1518), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1559), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1714), `activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1727), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1807), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1905), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2059), `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2121), `activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2162), `activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2184), `activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2204), `activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2231), `activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2265), `activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2312), `activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2528), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2588), `activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2597), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2646), `activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2676), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2722), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2755), `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2872), `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3006), `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3037), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3070), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3147), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3184), `activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3215), `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3262), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3334), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3354), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3418), `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3638), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3783), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3803), `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3929), and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3966).

```
6.301.3.5 virtual int activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1
( OpenWireFormat * format, commands::DataStructure * command,
  utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [pure
  virtual]
```

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>bs</i>	- boolean stream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 227), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 310), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 351), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 377), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 423), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 467), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 530), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 557), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 585), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 617), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 646), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 674), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 747), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 842), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 873), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1252), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1284), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1345), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1388), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1416), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1449), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1477), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1511), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1576), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1710), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1743), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1827), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1922), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2075), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2142), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2170),

activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 2193),  
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 2224),  
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 2251),  
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 2285),  
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 2333),  
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2545),  
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2585),  
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller  
(p. 2614), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2650),  
activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2672),  
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2718),  
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2772),  
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2894),  
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 3010),  
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 3041),  
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 3058),  
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 3156),  
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 3172),  
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 3203),  
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 3258),  
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 3347), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller  
(p. 3362), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller  
(p. 3426), activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3627),  
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3768),  
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3941),  
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3979),  
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 192),  
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller  
(p. 243), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller  
(p. 322), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller  
(p. 363), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller  
(p. 389), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller  
(p. 435), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller  
(p. 479), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller  
(p. 542), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller  
(p. 568), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller  
(p. 597), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller  
(p. 625), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller  
(p. 658), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller  
(p. 686), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller  
(p. 767), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 854),  
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 885),  
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1264),  
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1272),  
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1303),  
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1333),  
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1376),  
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1404),  
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1437), ac-

`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1465),  
`activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1498),  
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1564),  
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1698),  
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1731),  
`activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1811),  
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1910),  
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 2063),  
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 2126),  
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 2154),  
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 2177),  
`activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 2208),  
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 2235),  
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 2273),  
`activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 2317),  
`activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2533), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2569),  
`activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2602), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2630),  
`activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2664), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2702), `activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2752),  
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2877),  
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2990),  
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 3021), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 3054), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 3144), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 3180),  
`activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 3207),  
`activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 3244), `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 3327), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 3422), `activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 3643), `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3772),  
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3812),  
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 3933),  
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3971),  
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 180),  
`activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 223), `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 306), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 347), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 419), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 463), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 526), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 553), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 581), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 609), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 638), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`

(p. 666), `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 733), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 834), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 865), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1244), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1276), `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1307), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1337), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1380), `activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1441), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1469), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1503), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1568), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1702), `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1735), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1815), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1914), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2067), `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2134), `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2158), `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2181), `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2212), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2239), `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 2269), `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2321), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2537), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2573), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2606), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2642), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2659), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2710), `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2764), `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2886), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2998), `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3029), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3066), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3152), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3176), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3211), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3253), `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3343), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 3366), `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` (p. 3434), `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3623), `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3776), `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller` (p. 3945), `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3983), `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`

(p. 314), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`  
(p. 355), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`  
(p. 381), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`  
(p. 427), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`  
(p. 471), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`  
(p. 534), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`  
(p. 561), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`  
(p. 589), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`  
(p. 613), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`  
(p. 642), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`  
(p. 670), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`  
(p. 740), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 838),  
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 869), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1248),  
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1280),  
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1311),  
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1341),  
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1384),  
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1412), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1445), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1473),  
`activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1507),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1572),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1706),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1739),  
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1823),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1918),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2071),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2138),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2166),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2189),  
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2220),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2243),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2281),  
`activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2329),  
`activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2541), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2581),  
`activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`  
(p. 2610), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2634),  
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2668), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2714), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2768),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2890),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2994),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3025), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3050), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3164), `ac-`  
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3192),  
`activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3199),  
`activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3239), `ac-`

tivemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 3331), activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3374), activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 3438), activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 3635), activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (p. 3780), activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3808), activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3937), activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3975), activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (p. 196), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (p. 235), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (p. 318), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (p. 359), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (p. 385), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (p. 431), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (p. 475), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (p. 538), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller (p. 564), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (p. 593), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (p. 621), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (p. 650), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (p. 678), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller (p. 754), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 846), activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 877), activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1256), activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1288), activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1319), activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1349), activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1392), activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1420), activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1453), activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1481), activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1515), activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1555), activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1718), activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1747), activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1819), activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1926), activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 2079), activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 2130), activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 2150), activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 2197), activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 2216), activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2247), activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (p. 2277), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (p. 2325), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2549), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2577), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (p. 2618), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2638), activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2655), ac-

`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2706), `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2760), `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2881), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3002), `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3033), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3062), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3160), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3188), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3219), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3248), `tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3339), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3430), `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3631), `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3765), `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3792), `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3925), `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3987), `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 239), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 326), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 367), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 439), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller` (p. 483), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 546), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller` (p. 572), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller` (p. 601), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller` (p. 629), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 654), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller` (p. 682), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 760), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 850), `activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 881), `activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1260), `activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1292), `activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1323), `activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1353), `activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1396), `activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1424), `activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1457), `activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1485), `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1519), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1559), `activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1714), `activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1727), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1807), `activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1906), `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2059), `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2122), `activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2162),



activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller (p. 2185),  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller (p. 2204),  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller (p. 2231),  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller (p. 2265),  
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller (p. 2313),  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller (p. 2529),  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller (p. 2589),  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller  
 (p. 2597), activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller (p. 2646),  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller (p. 2677),  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller (p. 2722),  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller (p. 2756),  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller (p. 2873),  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller (p. 3006),  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller (p. 3037),  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller (p. 3070),  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller (p. 3148),  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller (p. 3184),  
 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller (p. 3215),  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller (p. 3262),  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller (p. 3335), activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller  
 (p. 3354), activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller  
 (p. 3418), activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller (p. 3639),  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller (p. 3783),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3804),  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller (p. 3929),  
 and activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller (p. 3967).

6.301.3.6 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2  
 ( OpenWireFormat \* *format*, commands::DataStructure \* *command*,  
 decaf::io::DataOutputStream \* *ds*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [pure virtual]

Tight Marshal to the given stream.

#### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Marshal
<i>ds</i>	- the DataOutputStream to Marshal to
<i>bs</i>	- boolean stream to marshal to.

#### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**  
 (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**  
 (p. 227), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**  
 (p. 311), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**

(p. 351), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`  
(p. 378), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`  
(p. 424), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller`  
(p. 467), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`  
(p. 530), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`  
(p. 557), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller`  
(p. 585), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`  
(p. 618), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`  
(p. 647), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller`  
(p. 675), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`  
(p. 748), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 843),  
`activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 874), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1253),  
`activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1285),  
`activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1316),  
`activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1346),  
`activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1389),  
`activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1417), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1450), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1478),  
`activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1511),  
`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1576),  
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1711),  
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1744),  
`activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1828),  
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1922),  
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2076),  
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2142),  
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2171),  
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2193),  
`activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2224),  
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2252),  
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2286),  
`activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2333),  
`activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2545), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2585),  
`activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller`  
(p. 2615), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2651),  
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2673), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2719), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2772),  
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2895),  
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3011),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3042), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3059), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3156), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3173),  
`activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3204),  
`activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3258), `ac-`  
`ativemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3347), `activemq::wireformat::openwi`

(p. 3363), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**  
(p. 3427), **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**  
(p. 3627), **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3769),  
**activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 3796),  
**activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller** (p. 3942),  
**activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3979),  
**activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 193),  
**activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**  
(p. 243), **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**  
(p. 323), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**  
(p. 363), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**  
(p. 390), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**  
(p. 436), **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**  
(p. 479), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**  
(p. 542), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**  
(p. 569), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**  
(p. 597), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**  
(p. 626), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**  
(p. 659), **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**  
(p. 687), **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**  
(p. 768), **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller** (p. 855),  
**activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 886), **ac-**  
**tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 1265),  
**activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 1273),  
**activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller** (p. 1304),  
**activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1334),  
**activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1377),  
**activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller** (p. 1405), **ac-**  
**tivemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1438), **ac-**  
**tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1466),  
**activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1499),  
**activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1564),  
**activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1699),  
**activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller** (p. 1732),  
**activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1812),  
**activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1910),  
**activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2064),  
**activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller** (p. 2126),  
**activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller** (p. 2155),  
**activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller** (p. 2177),  
**activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller** (p. 2208),  
**activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 2236),  
**activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2274),  
**activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2317),  
**activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 2533), **ac-**  
**tivemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 2569),  
**activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**  
(p. 2602), **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller** (p. 2631),  
**activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2664), **ac-**  
**tivemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 2703), **ac-**

tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2752),  
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2877),  
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2991),  
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3022), ac-  
tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3055), ac-  
tivemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3144), ac-  
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 3181),  
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 3208),  
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3244), ac-  
tivemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3327), activemq::wireformat::openw-  
(p. 3371), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller  
(p. 3423), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller  
(p. 3643), activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3773),  
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3812),  
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3934),  
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3971),  
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 181),  
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller  
(p. 223), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller  
(p. 307), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller  
(p. 347), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller  
(p. 374), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller  
(p. 420), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller  
(p. 463), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller  
(p. 526), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller  
(p. 554), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller  
(p. 581), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller  
(p. 610), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller  
(p. 639), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller  
(p. 667), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller  
(p. 734), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 835),  
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 866), ac-  
tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1245),  
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1277),  
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1308),  
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1338),  
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1381),  
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1409), ac-  
tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1442), ac-  
tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1470),  
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1503),  
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1568),  
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1703),  
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1736),  
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1816),  
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1914),  
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 2068),  
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 2134),  
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 2159),  
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 2181),

activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 2212),  
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 2240),  
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 2270),  
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 2321),  
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2537), ac-  
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2573),  
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller  
(p. 2606), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2643),  
activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2660), ac-  
tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2711), ac-  
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2764),  
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2886),  
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2999),  
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 3030), ac-  
tivemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 3067), ac-  
tivemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 3152), ac-  
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 3177),  
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 3212),  
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 3254), ac-  
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 3343), activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller  
(p. 3367), activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller  
(p. 3435), activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3623), activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3776),  
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3800),  
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3946),  
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3983),  
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (p. 189),  
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller  
(p. 231), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller  
(p. 315), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller  
(p. 355), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller  
(p. 382), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller  
(p. 428), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller  
(p. 471), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller  
(p. 534), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller  
(p. 561), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller  
(p. 589), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller  
(p. 614), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller  
(p. 643), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller  
(p. 671), activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller  
(p. 741), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 839),  
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 870), ac-  
tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1249),  
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1281),  
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1312),  
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1342),  
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1385),  
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1413), ac-  
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1446), ac-  
tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1474),

`activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1507),  
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1572),  
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1707),  
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1740),  
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1824),  
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1918),  
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 2072),  
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 2138),  
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 2167),  
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 2189),  
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 2220),  
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 2244),  
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 2282),  
`activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2329),  
`activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2541), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2581),  
`activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2610), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2635),  
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2669), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2715), `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2768),  
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2890),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2995),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3026), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3051), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3164), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3193),  
`activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3200),  
`activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3240), `activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3331), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3439), `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3635), `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3780),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3808),  
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3938),  
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3975),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 197),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 235), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 319),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 359), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 386),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 432), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 475),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 538), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 565),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 593), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 622),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 651), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 679), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`

(p. 755), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 847), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 878), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1257), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1289), `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1320), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1350), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1393), `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1421), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1454), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1482), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1515), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1556), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1719), `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1748), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1820), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1926), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2080), `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2130), `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2151), `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2197), `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2216), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 2248), `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 2278), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2325), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2549), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2577), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2619), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2639), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2656), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2707), `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2760), `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2882), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 3003), `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 3034), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 3063), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 3160), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 3189), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 3220), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 3249), `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 3339), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 3359), `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 3431), `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3631), `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3765), `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3792), `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3926), `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3987), `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 201), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 239), `activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller` (p. 327), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller`

(p. 367), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller`  
(p. 394), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller`  
(p. 440), `activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller`  
(p. 483), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller`  
(p. 546), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller`  
(p. 573), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller`  
(p. 601), `activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller`  
(p. 630), `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller`  
(p. 655), `activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller`  
(p. 683), `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller`  
(p. 762), `activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller` (p. 851),  
`activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller` (p. 882), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller` (p. 1261),  
`activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller` (p. 1293),  
`activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller` (p. 1324),  
`activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller` (p. 1354),  
`activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller` (p. 1397),  
`activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller` (p. 1425), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller` (p. 1458), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller` (p. 1486),  
`activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1519),  
`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1560),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1715),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1728),  
`activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1808),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1906),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2060),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2122),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2163),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2185),  
`activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2204),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2232),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2266),  
`activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2313),  
`activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2529), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2589),  
`activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller`  
(p. 2598), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2647),  
`activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2677), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2723), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2756),  
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2873),  
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3007),  
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3038), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3071), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3148), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3185),  
`activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3216),  
`activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3263), `ac-`  
`tivemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3335), `activemq::wireformat::openwi`



## 6.301 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference 1627

(p. 3355), [activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller](#) (p. 3419), [activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller](#) (p. 3639), [activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller](#) (p. 3784), [activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller](#) (p. 3804), [activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller](#) (p. 3930), and [activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller](#) (p. 3967).

```
6.301.3.7 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal
( OpenWireFormat * format, commands::DataStructure * command,
  decaf::io::DataInputStream * dis, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [pure virtual]
```

Tight Un-marhsal to the given stream.

### Parameters

<i>format</i>	- The OpenWireFormat properties
<i>command</i>	- the object to Un-Marshal
<i>dis</i>	- the DataInputStream to Un-Marshal from
<i>bs</i>	- boolean stream to unmarshal from.

### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

Implemented in [activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller](#) (p. 185), [activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller](#) (p. 228), [activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller](#) (p. 311), [activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller](#) (p. 352), [activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller](#) (p. 378), [activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller](#) (p. 424), [activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller](#) (p. 468), [activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller](#) (p. 531), [activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller](#) (p. 558), [activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller](#) (p. 586), [activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller](#) (p. 618), [activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller](#) (p. 647), [activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller](#) (p. 675), [activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller](#) (p. 749), [activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller](#) (p. 843), [activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller](#) (p. 874), [activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller](#) (p. 1253), [activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller](#) (p. 1285), [activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller](#) (p. 1316), [activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller](#) (p. 1346), [activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller](#) (p. 1389), [activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller](#) (p. 1417), [activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller](#) (p. 1450), [activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller](#) (p. 1478), [activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller](#) (p. 1512),

`activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1577),  
`activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1711),  
`activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1744),  
`activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1828),  
`activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1923),  
`activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2076),  
`activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 2143),  
`activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 2171),  
`activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller` (p. 2194),  
`activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller` (p. 2225),  
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 2252),  
`activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2286),  
`activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 2334),  
`activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2546), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2586),  
`activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2615), `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller` (p. 2651),  
`activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2674), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2719), `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller` (p. 2773),  
`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2895),  
`activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 3011),  
`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller` (p. 3042), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 3059), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 3157), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 3173),  
`activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 3204),  
`activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 3259), `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller` (p. 3348), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 3363), `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller` (p. 3427),  
`activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3628), `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3769),  
`activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 3797),  
`activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller` (p. 3942),  
`activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3980),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 193),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 244), `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 323),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 364), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 390),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 436), `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 480),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 543), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 569),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 598), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 626),  
`activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 659), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 687),  
`activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 769), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 855),

activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 886), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1265), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1273), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1304), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1334), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1377), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1405), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1438), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1466), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1499), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1565), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1699), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1732), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1812), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1911), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 2064), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 2127), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 2155), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 2178), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 2209), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 2236), activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 2274), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 2318), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2534), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2570), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 2603), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2631), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2665), activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2703), activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2753), activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2878), activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2991), activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 3022), activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 3055), activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 3145), activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 3181), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 3208), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 3245), activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 3328), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 3371), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3423), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 3644), activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3773), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3813), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3934), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3972), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 181), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 224), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (p. 307), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 348), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller

(p. 374), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`  
(p. 420), `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller`  
(p. 464), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`  
(p. 527), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller`  
(p. 554), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`  
(p. 582), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`  
(p. 610), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`  
(p. 639), `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`  
(p. 667), `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`  
(p. 736), `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller` (p. 835),  
`activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 866), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1245),  
`activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1277),  
`activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller` (p. 1308),  
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1338),  
`activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1381),  
`activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller` (p. 1409), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1442), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1470),  
`activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1504),  
`activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1569),  
`activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1703),  
`activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller` (p. 1736),  
`activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1816),  
`activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1915),  
`activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 2068),  
`activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller` (p. 2135),  
`activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller` (p. 2159),  
`activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller` (p. 2182),  
`activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller` (p. 2213),  
`activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 2240),  
`activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 2270),  
`activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 2322),  
`activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2538), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2574),  
`activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`  
(p. 2607), `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller` (p. 2643),  
`activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2661), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2711), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller` (p. 2765),  
`activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2887),  
`activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2999),  
`activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` (p. 3030), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 3067), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 3153), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 3177),  
`activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 3212),  
`activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 3254), `ac-`  
`tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller` (p. 3344), `activemq::wireformat::openwi-`  
(p. 3367), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller`

(p. 3435), **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**  
(p. 3624), **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3777),  
**activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3801),  
**activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 3946),  
**activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3984),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 189),  
**activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**  
(p. 232), **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**  
(p. 315), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**  
(p. 356), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**  
(p. 382), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**  
(p. 428), **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**  
(p. 472), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**  
(p. 535), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**  
(p. 562), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**  
(p. 590), **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**  
(p. 614), **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**  
(p. 643), **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**  
(p. 671), **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**  
(p. 742), **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller** (p. 839),  
**activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller** (p. 870), **ac-**  
**tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller** (p. 1249),  
**activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller** (p. 1281),  
**activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller** (p. 1312),  
**activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller** (p. 1342),  
**activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller** (p. 1385),  
**activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller** (p. 1413), **ac-**  
**tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller** (p. 1446), **ac-**  
**tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller** (p. 1474),  
**activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1508),  
**activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1573),  
**activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller** (p. 1707),  
**activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller** (p. 1740),  
**activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1824),  
**activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller** (p. 1919),  
**activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2072),  
**activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller** (p. 2139),  
**activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller** (p. 2167),  
**activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller** (p. 2190),  
**activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller** (p. 2221),  
**activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller** (p. 2244),  
**activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2282),  
**activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2330),  
**activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller** (p. 2542), **ac-**  
**tivemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller** (p. 2582),  
**activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**  
(p. 2611), **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller** (p. 2635),  
**activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2669), **ac-**  
**tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller** (p. 2715), **ac-**  
**tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller** (p. 2769),

`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2891),  
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2995),  
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 3026), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 3051), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 3165), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 3193),  
`activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 3200),  
`activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 3240), `activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 3332), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 3439), `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 3636), `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3781),  
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3809),  
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3938),  
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3976),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 197),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 236), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 319),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 360), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 386),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 432), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 476),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 539), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 566),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 594), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 622),  
`activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 651), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 679),  
`activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 756), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 847),  
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 878), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1257),  
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1289),  
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1320),  
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1350),  
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1393),  
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1421), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1454), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1482),  
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1516),  
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1556),  
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1719),  
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1748),  
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1820),  
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1927),  
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 2080),  
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 2131),  
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 2151),  
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 2198),  
`activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 2217),

activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 2248),  
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (p. 2278),  
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (p. 2326),  
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2550), ac-  
tivemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2578),  
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller  
(p. 2619), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2639),  
activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2656), ac-  
tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2707), ac-  
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2761),  
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2882),  
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 3003),  
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 3034), ac-  
tivemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 3063), ac-  
tivemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 3161), ac-  
tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (p. 3189),  
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (p. 3220),  
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 3250), ac-  
tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 3340), activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller  
(p. 3359), activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller  
(p. 3431), activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3766),  
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3793),  
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3926),  
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3988),  
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller (p. 201),  
activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller  
(p. 240), activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller  
(p. 327), activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller  
(p. 368), activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller  
(p. 394), activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller  
(p. 440), activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller  
(p. 484), activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller  
(p. 547), activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller  
(p. 573), activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller  
(p. 602), activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller  
(p. 630), activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller  
(p. 655), activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller  
(p. 683), activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller  
(p. 763), activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller (p. 851),  
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller (p. 882), ac-  
tivemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller (p. 1261),  
activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller (p. 1293),  
activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller (p. 1324),  
activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller (p. 1354),  
activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller (p. 1397),  
activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller (p. 1425), ac-  
tivemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller (p. 1458), ac-  
tivemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller (p. 1486),  
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller (p. 1520),

`activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1560),  
`activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller` (p. 1715),  
`activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller` (p. 1728),  
`activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1808),  
`activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller` (p. 1907),  
`activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2060),  
`activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` (p. 2123),  
`activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller` (p. 2163),  
`activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller` (p. 2186),  
`activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller` (p. 2205),  
`activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller` (p. 2232),  
`activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller` (p. 2266),  
`activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller` (p. 2314),  
`activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller` (p. 2530), `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller` (p. 2590),  
`activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller` (p. 2598), `activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller` (p. 2647),  
`activemq::wireformat::openwire::marshal::v6::MessageMarshaller` (p. 2678), `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller` (p. 2723), `activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller` (p. 2757),  
`activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller` (p. 2874),  
`activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller` (p. 3007),  
`activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller` (p. 3038), `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller` (p. 3071), `activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller` (p. 3149), `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller` (p. 3185),  
`activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller` (p. 3216),  
`activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3264), `activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller` (p. 3336), `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller` (p. 3355), `activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller` (p. 3419), `activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller` (p. 3640), `activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3784),  
`activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller` (p. 3805),  
`activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller` (p. 3930),  
and `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller` (p. 3968).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h`

## 6.302 `activemq::commands::DataStructure` Class Reference

```
#include <src/main/activemq/commands/DataStructure.h>
```

Inheritance diagram for `activemq::commands::DataStructure`:



## Public Member Functions

- virtual `~DataStructure ()`
- virtual unsigned char `getDataStructureType ()` const =0  
*Get the **DataStructure** (p. 1628) Type as defined in CommandTypes.h.*
- virtual `DataStructure * cloneDataStructure ()` const =0  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void `copyDataStructure (const DataStructure *src)=0`  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string `toString ()` const =0  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool `equals (const DataStructure *value)` const =0  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*

## 6.302.1 Constructor & Destructor Documentation

6.302.1.1 virtual `activemq::commands::DataStructure::~~DataStructure ( )` [inline, virtual]

## 6.302.2 Member Function Documentation

6.302.2.1 virtual `DataStructure* activemq::commands::DataStructure::cloneDataStructure ( )` const [pure virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 174), `activemq::commands::ActiveMQBytesMessage` (p. 205), `activemq::commands::ActiveMQDestination` (p. 296), `activemq::commands::ActiveMQMapMessage` (p. 334), `activemq::commands::ActiveMQMessage` (p. 369), `activemq::commands::ActiveMQObjectMessage` (p. 415), `activemq::commands::ActiveMQQueue` (p. 454), `activemq::commands::ActiveMQStreamMessage` (p. 510), `activemq::commands::ActiveMQTempDestination` (p. 548), `activemq::commands::ActiveMQTempQueue` (p. 575), `activemq::commands::ActiveMQTempTopic` (p. 604), `activemq::commands::ActiveMQTextMessage` (p. 633), `activemq::commands::ActiveMQTopic` (p. 661), `activemq::commands::BooleanExpression` (p. 817), `activemq::commands::BrokerError` (p. 824), `activemq::commands::BrokerId` (p. 830), `activemq::commands::BrokerInfo` (p. 858), `activemq::commands::ConnectionControl` (p. 1238), `activemq::commands::ConnectionError` (p. 1267), `activemq::commands::ConnectionId` (p. 1298), `activemq::commands::ConnectionInfo` (p. 1326), `activemq::commands::ConsumerControl` (p. 1370), `activemq::commands::ConsumerId` (p. 1399), `activemq::commands::ConsumerInfo`

(p. 1428), `activemq::commands::ControlCommand` (p. 1460), `activemq::commands::DataArrayResponse` (p. 1494), `activemq::commands::DataResponse` (p. 1551), `activemq::commands::DestinationInfo` (p. 1693), `activemq::commands::DiscoveryEvent` (p. 1723), `activemq::commands::ExceptionResponse` (p. 1803), `activemq::commands::FlushCommand` (p. 1901), `activemq::commands::IntegerResponse` (p. 2055), `activemq::commands::JournalQueueAck` (p. 2117), `activemq::commands::JournalTopicAck` (p. 2145), `activemq::commands::JournalTrace` (p. 2173), `activemq::commands::JournalTransaction` (p. 2199), `activemq::commands::KeepAliveInfo` (p. 2226), `activemq::commands::LastPartialCommand` (p. 2261), `activemq::commands::LocalTransactionId` (p. 2308), `activemq::commands::Message` (p. 2480), `activemq::commands::MessageAck` (p. 2522), `activemq::commands::MessageDispatch` (p. 2556), `activemq::commands::MessageDispatchNotification` (p. 2592), `activemq::commands::MessagePull` (p. 2625), `activemq::commands::MessagePull` (p. 2697), `activemq::commands::NetworkBridgeFilter` (p. 2747), `activemq::commands::PartialCommand` (p. 2868), `activemq::commands::ProducerAck` (p. 2985), `activemq::commands::ProducerId` (p. 3016), `activemq::commands::ProducerInfo` (p. 3044), `activemq::commands::RemoveInfo` (p. 3139), `activemq::commands::RemoveSubscriptionInfo` (p. 3166), `activemq::commands::ReplayCommand` (p. 3195), `activemq::commands::Response` (p. 3228), `activemq::commands::SessionId` (p. 3322), `activemq::commands::SessionInfo` (p. 3349), `activemq::commands::ShutdownInfo` (p. 3414), `activemq::commands::SubscriptionInfo` (p. 3617), `activemq::commands::TransactionId` (p. 3761), `activemq::commands::TransactionInfo` (p. 3786), `activemq::commands::WireFormatInfo` (p. 3915), and `activemq::commands::XATransactionId` (p. 3962).

```
6.302.2.2  virtual void activemq::commands::DataStructure::copyDataStructure ( const
           DataStructure * src ) [pure virtual]
```

Copy the contents of the passed object into this objects members, overwriting any existing data.

## Returns

src - Source Object

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 174), `activemq::commands::ActiveMQBMessage` (p. 205), `activemq::commands::ActiveMQDestination` (p. 296), `activemq::commands::ActiveMQMapMessage` (p. 334), `activemq::commands::ActiveMQMessage` (p. 370), `activemq::commands::ActiveMQObjectMessage` (p. 415), `activemq::commands::ActiveMQQueue` (p. 455), `activemq::commands::ActiveMQStreamMessage` (p. 510), `activemq::commands::ActiveMQTempDestination` (p. 549), `activemq::commands::ActiveMQTempMessage` (p. 576), `activemq::commands::ActiveMQTempTopic` (p. 604), `activemq::commands::ActiveMQTextMessage` (p. 633), `activemq::commands::ActiveMQTopic` (p. 662), `activemq::commands::BaseCommand` (p. 724), `activemq::commands::BrokerError` (p. 825), `activemq::commands::BrokerId` (p. 830), `activemq::commands::BrokerInfo` (p. 858), `activemq::commands::ConnectionControl` (p. 1239), `activemq::commands::ConnectionError` (p. 1267), `activemq::commands::ConnectionId` (p. 1299), `activemq::commands::ConnectionInfo` (p. 1327), `activemq::commands::ConsumerControl` (p. 1371), `activemq::commands::ConsumerId` (p. 1400), `activemq::commands::ConsumerInfo` (p. 1429), `activemq::commands::ControlCommand` (p. 1461), `activemq::commands::DataArrayResponse` (p. 1494), `activemq::commands::DataResponse` (p. 1551), `activemq::commands::DestinationInfo` (p. 1693), `activemq::commands::DiscoveryEvent` (p. 1723), `activemq::commands::ExceptionResponse` (p. 1803), `activemq::commands::FlushCommand` (p. 1902), `activemq::commands::IntegerResponse` (p. 2055), `activemq::commands::JournalQueueAck` (p. 2117), `activemq::commands::JournalTopicAck` (p. 2145), `activemq::commands::JournalTrace` (p. 2173), `activemq::commands::JournalTransaction`

(p. 2200), [activemq::commands::KeepAliveInfo](#) (p. 2227), [activemq::commands::LastPartialCommand](#) (p. 2261), [activemq::commands::LocalTransactionId](#) (p. 2308), [activemq::commands::Message](#) (p. 2481), [activemq::commands::MessageAck](#) (p. 2522), [activemq::commands::MessageDispatch](#) (p. 2556), [activemq::commands::MessageDispatchNotification](#) (p. 2592), [activemq::commands::MessageId](#) (p. 2626), [activemq::commands::MessagePull](#) (p. 2697), [activemq::commands::NetworkBridgeFilter](#) (p. 2747), [activemq::commands::PartialCommand](#) (p. 2868), [activemq::commands::ProducerAck](#) (p. 2985), [activemq::commands::ProducerId](#) (p. 3016), [activemq::commands::ProducerInfo](#) (p. 3044), [activemq::commands::RemoveInfo](#) (p. 3139), [activemq::commands::RemoveSubscriptionInfo](#) (p. 3167), [activemq::commands::ReplayCommand](#) (p. 3195), [activemq::commands::Response](#) (p. 3229), [activemq::commands::SessionId](#) (p. 3322), [activemq::commands::SessionInfo](#) (p. 3350), [activemq::commands::ShutdownInfo](#) (p. 3414), [activemq::commands::SubscriptionInfo](#) (p. 3618), [activemq::commands::TransactionId](#) (p. 3761), [activemq::commands::TransactionInfo](#) (p. 3787), [activemq::commands::WireFormatInfo](#) (p. 3915), and [activemq::commands::XATransactionId](#) (p. 3962).

```
6.302.2.3 virtual bool activemq::commands::DataStructure::equals ( const DataStructure *
value ) const [pure virtual]
```

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Implemented in [activemq::commands::ActiveMQBlobMessage](#) (p. 175), [activemq::commands::ActiveMQBytesMessage](#) (p. 206), [activemq::commands::ActiveMQDestination](#) (p. 297), [activemq::commands::ActiveMQMapMessage](#) (p. 334), [activemq::commands::ActiveMQMessage](#) (p. 370), [activemq::commands::ActiveMQMessageTemplate< T >](#) (p. 399), [activemq::commands::ActiveMQObjectMessage](#) (p. 415), [activemq::commands::ActiveMQQueue](#) (p. 455), [activemq::commands::ActiveMQStreamMessage](#) (p. 510), [activemq::commands::ActiveMQTempDestination](#) (p. 549), [activemq::commands::ActiveMQTempQueue](#) (p. 576), [activemq::commands::ActiveMQTempTopic](#) (p. 605), [activemq::commands::ActiveMQTextMessage](#) (p. 633), [activemq::commands::ActiveMQTopic](#) (p. 662), [activemq::commands::BaseCommand](#) (p. 725), [activemq::commands::BooleanExpression](#) (p. 817), [activemq::commands::BrokerId](#) (p. 830), [activemq::commands::BrokerInfo](#) (p. 858), [activemq::commands::ConnectionControl](#) (p. 1239), [activemq::commands::ConnectionError](#) (p. 1268), [activemq::commands::ConnectionId](#) (p. 1299), [activemq::commands::ConnectionInfo](#) (p. 1327), [activemq::commands::ConsumerControl](#) (p. 1371), [activemq::commands::ConsumerId](#) (p. 1400), [activemq::commands::ConsumerInfo](#) (p. 1429), [activemq::commands::ControlCommand](#) (p. 1461), [activemq::commands::DataArrayResponse](#) (p. 1495), [activemq::commands::DataResponse](#) (p. 1551), [activemq::commands::DestinationInfo](#) (p. 1693), [activemq::commands::DiscoveryEvent](#) (p. 1723), [activemq::commands::ExceptionResponse](#) (p. 1803), [activemq::commands::FlushCommand](#) (p. 1902), [activemq::commands::IntegerResponse](#) (p. 2055), [activemq::commands::JournalQueueAck](#) (p. 2118), [activemq::commands::JournalTopicAck](#) (p. 2145), [activemq::commands::JournalTrace](#) (p. 2173), [activemq::commands::JournalTransaction](#) (p. 2200), [activemq::commands::KeepAliveInfo](#) (p. 2227), [activemq::commands::LastPartialCommand](#) (p. 2261), [activemq::commands::LocalTransactionId](#) (p. 2309), [activemq::commands::Message](#) (p. 2481), [activemq::commands::MessageAck](#) (p. 2523), [activemq::commands::MessageDispatch](#) (p. 2557), [activemq::commands::MessageDispatchNotification](#)

(p. 2592), `activemq::commands::MessageId` (p. 2626), `activemq::commands::MessagePull` (p. 2697), `activemq::commands::NetworkBridgeFilter` (p. 2748), `activemq::commands::PartialCommand` (p. 2868), `activemq::commands::ProducerAck` (p. 2986), `activemq::commands::ProducerId` (p. 3017), `activemq::commands::ProducerInfo` (p. 3045), `activemq::commands::RemoveInfo` (p. 3139), `activemq::commands::RemoveSubscriptionInfo` (p. 3167), `activemq::commands::ReplayCommand` (p. 3195), `activemq::commands::Response` (p. 3229), `activemq::commands::SessionId` (p. 3322), `activemq::commands::SessionInfo` (p. 3350), `activemq::commands::ShutdownInfo` (p. 3414), `activemq::commands::SubscriptionInfo` (p. 3618), `activemq::commands::TransactionId` (p. 3761), `activemq::commands::TransactionInfo` (p. 3787), `activemq::commands::WireFormatInfo` (p. 3915), `activemq::commands::XATransactionId` (p. 3962), `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 399), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 399).

6.302.2.4 `virtual unsigned char activemq::commands::DataStructure::getDataStructureType ( )`  
`const [pure virtual]`

Get the **DataStructure** (p. 1628) Type as defined in `CommandTypes.h`.

## Returns

The type of the data structure

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 175), `activemq::commands::ActiveMQBlobMessage` (p. 207), `activemq::commands::ActiveMQDestination` (p. 298), `activemq::commands::ActiveMQMapMessage` (p. 336), `activemq::commands::ActiveMQMessage` (p. 370), `activemq::commands::ActiveMQObjectMessage` (p. 416), `activemq::commands::ActiveMQQueue` (p. 456), `activemq::commands::ActiveMQStreamMessage` (p. 510), `activemq::commands::ActiveMQTempDestination` (p. 550), `activemq::commands::ActiveMQTempDestination` (p. 577), `activemq::commands::ActiveMQTempTopic` (p. 605), `activemq::commands::ActiveMQTextMessage` (p. 633), `activemq::commands::ActiveMQTopic` (p. 663), `activemq::commands::BrokerError` (p. 825), `activemq::commands::BrokerId` (p. 831), `activemq::commands::BrokerInfo` (p. 859), `activemq::commands::ConnectionControl` (p. 1239), `activemq::commands::ConnectionError` (p. 1268), `activemq::commands::ConnectionId` (p. 1299), `activemq::commands::ConnectionInfo` (p. 1328), `activemq::commands::ConsumerControl` (p. 1371), `activemq::commands::ConsumerId` (p. 1400), `activemq::commands::ConsumerInfo` (p. 1430), `activemq::commands::ControlCommand` (p. 1461), `activemq::commands::DataArrayResponse` (p. 1495), `activemq::commands::DataResponse` (p. 1552), `activemq::commands::DestinationInfo` (p. 1694), `activemq::commands::DiscoveryEvent` (p. 1724), `activemq::commands::ExceptionResponse` (p. 1803), `activemq::commands::FlushCommand` (p. 1902), `activemq::commands::IntegerResponse` (p. 2056), `activemq::commands::JournalQueueAck` (p. 2118), `activemq::commands::JournalTopicAck` (p. 2145), `activemq::commands::JournalTrace` (p. 2173), `activemq::commands::JournalTransaction` (p. 2200), `activemq::commands::KeepAliveInfo` (p. 2227), `activemq::commands::LastPartialCommand` (p. 2262), `activemq::commands::LocalTransactionId` (p. 2309), `activemq::commands::Message` (p. 2483), `activemq::commands::MessageAck` (p. 2523), `activemq::commands::MessageDispatch` (p. 2557), `activemq::commands::MessageDispatchNotAck` (p. 2593), `activemq::commands::MessageId` (p. 2626), `activemq::commands::MessagePull` (p. 2698), `activemq::commands::NetworkBridgeFilter` (p. 2748), `activemq::commands::PartialCommand`

(p. 2869), **activemq::commands::ProducerAck** (p. 2986), **activemq::commands::ProducerId** (p. 3017), **activemq::commands::ProducerInfo** (p. 3045), **activemq::commands::RemoveInfo** (p. 3139), **activemq::commands::RemoveSubscriptionInfo** (p. 3168), **activemq::commands::ReplayCommand** (p. 3196), **activemq::commands::Response** (p. 3230), **activemq::commands::SessionId** (p. 3323), **activemq::commands::SessionInfo** (p. 3350), **activemq::commands::ShutdownInfo** (p. 3415), **activemq::commands::SubscriptionInfo** (p. 3618), **activemq::commands::TransactionId** (p. 3762), **activemq::commands::TransactionInfo** (p. 3787), **activemq::commands::WireFormatInfo** (p. 3916), and **activemq::commands::XATransactionId** (p. 3963).

**6.302.2.5** `virtual std::string activemq::commands::DataStructure::toString ( ) const [pure virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 177), **activemq::commands::ActiveMQBytesMessage** (p. 214), **activemq::commands::ActiveMQDestination** (p. 302), **activemq::commands::ActiveMQMapMessage** (p. 344), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 416), **activemq::commands::ActiveMQQueue** (p. 457), **activemq::commands::ActiveMQStreamMessage** (p. 518), **activemq::commands::ActiveMQTempDestination** (p. 550), **activemq::commands::ActiveMQTempQueue** (p. 578), **activemq::commands::ActiveMQTempTopic** (p. 606), **activemq::commands::ActiveMQTextMessage** (p. 635), **activemq::commands::ActiveMQTopic** (p. 663), **activemq::commands::BaseCommand** (p. 729), **activemq::commands::BaseDataStructure** (p. 796), **activemq::commands::BooleanExpression** (p. 817), **activemq::commands::BrokerId** (p. 831), **activemq::commands::BrokerInfo** (p. 861), **activemq::commands::Command** (p. 1169), **activemq::commands::ConnectionControl** (p. 1241), **activemq::commands::ConnectionError** (p. 1269), **activemq::commands::ConnectionId** (p. 1300), **activemq::commands::ConnectionInfo** (p. 1329), **activemq::commands::ConsumerControl** (p. 1372), **activemq::commands::ConsumerId** (p. 1401), **activemq::commands::ConsumerInfo** (p. 1432), **activemq::commands::ControlCommand** (p. 1462), **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse** (p. 1552), **activemq::commands::DestinationInfo** (p. 1695), **activemq::commands::DiscoveryEvent** (p. 1724), **activemq::commands::ExceptionResponse** (p. 1804), **activemq::commands::FlushCommand** (p. 1902), **activemq::commands::IntegerResponse** (p. 2056), **activemq::commands::JournalQueueAck** (p. 2119), **activemq::commands::JournalTopicAck** (p. 2147), **activemq::commands::JournalTrace** (p. 2174), **activemq::commands::JournalTransaction** (p. 2201), **activemq::commands::KeepAliveInfo** (p. 2228), **activemq::commands::LastPartialCommand** (p. 2262), **activemq::commands::LocalTransactionId** (p. 2310), **activemq::commands::Message** (p. 2490), **activemq::commands::MessageAck** (p. 2525), **activemq::commands::MessageDispatch** (p. 2558), **activemq::commands::MessageDispatchNotification** (p. 2594), **activemq::commands::MessageId** (p. 2627), **activemq::commands::MessagePull** (p. 2699), **activemq::commands::NetworkBridgeFilter** (p. 2748), **activemq::commands::PartialCommand** (p. 2869), **activemq::commands::ProducerAck** (p. 2987), **activemq::commands::ProducerId** (p. 3018), **activemq::commands::ProducerInfo** (p. 3046), **activemq::commands::RemoveInfo** (p. 3140), **activemq::commands::RemoveSubscriptionInfo** (p. 3168), **activemq::commands::ReplayCommand** (p. 3196), **activemq::commands::Response** (p. 3230), **activemq::commands::SessionId** (p. 3324), **activemq::commands::SessionInfo** (p. 3351), **activemq::commands::ShutdownInfo** (p. 3415), **activemq::commands::SubscriptionInfo**

(p. 3619), **activemq::commands::TransactionId** (p. 3762), **activemq::commands::TransactionInfo** (p. 3788), **activemq::commands::WireFormatInfo** (p. 3922), and **activemq::commands::XATransactionId** (p. 3964).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataStructure.h`

### 6.303 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

```
#include <src/main/decaf/util/Date.h>
```

Inheritance diagram for `decaf::util::Date`:

#### Public Member Functions

- **Date** ()  
*Default constructor - sets time to the current System time, rounded to the nearest millisecond.*
- **Date** (long long milliseconds)  
*Constructs the date with a given time value.*
- **Date** (const **Date** &source)  
*Copy constructor.*
- **Date & operator=** (const **Date** &value)  
*Assigns the value of one **Date** (p. 1633) object to another.*
- virtual **~Date** ()
- long long **getTime** () const  
*Gets the underlying time.*
- void **setTime** (long long milliseconds)  
*Sets the underlying time.*
- bool **after** (const **Date** &when) const  
*Determines whether or not this date falls after the specified time.*
- bool **before** (const **Date** &when) const  
*Determines whether or not this date falls before the specified time.*
- std::string **toString** () const  
*Converts this **Date** (p. 1633) object to a String of the form:*
- virtual int **compareTo** (const **Date** &value) const  
*Compares this Data object to the one given.*
- virtual bool **equals** (const **Date** &value) const
- virtual bool **operator==** (const **Date** &value) const  
*Compares equality between this object and the one passed.*

- virtual bool **operator**< (const **Date** &value) const

*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### 6.303.1 Detailed Description

Wrapper class around a time value in milliseconds.

This class is comparable to Java's java.util.Date class.

#### Since

1.0

### 6.303.2 Constructor & Destructor Documentation

#### 6.303.2.1 decaf::util::Date::Date ( )

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

#### 6.303.2.2 decaf::util::Date::Date ( long long *milliseconds* )

Constructs the date with a given time value.

#### Parameters

<i>milliseconds</i>	The time in milliseconds;
---------------------	---------------------------

#### 6.303.2.3 decaf::util::Date::Date ( const **Date** & *source* )

Copy constructor.

#### Parameters

<i>source</i>	The <b>Date</b> (p. 1633) instance to copy into this one.
---------------	---

#### 6.303.2.4 virtual decaf::util::Date::~~Date ( ) [virtual]

### 6.303.3 Member Function Documentation

#### 6.303.3.1 bool decaf::util::Date::after ( const **Date** & *when* ) const

Determines whether or not this date falls after the specified time.

**Parameters**

<i>when</i>	The date to compare
-------------	---------------------

**Returns**

true if this date falls after when.

**6.303.3.2 bool decaf::util::Date::before ( const Date & *when* ) const**

Determines whether or not this date falls before the specified time.

**Parameters**

<i>when</i>	The date to compare
-------------	---------------------

**Returns**

true if this date falls before when.

**6.303.3.3 virtual int decaf::util::Date::compareTo ( const Date & *value* ) const [virtual]**

Compares this Date object to the one given.

**Parameters**

<i>value</i>	The <b>Date</b> (p. 1633) value to compare to this one.
--------------	---

**Returns**

zero if the **Date** (p. 1633) values are equal, a value less than zero if this Date value is earlier than argument value, and a value greater than zero if this **Date** (p. 1633) object is later than the argument **Date** (p. 1633) value.

**6.303.3.4 virtual bool decaf::util::Date::equals ( const Date & *value* ) const [virtual]****Returns**

true if this value is considered equal to the passed value.

**6.303.3.5 long long decaf::util::Date::getTime ( ) const**

Gets the underlying time.

**Returns**

The underlying time value in milliseconds.



6.303.3.6 `virtual bool decaf::util::Date::operator< ( const Date & value ) const`  
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

6.303.3.7 `Date& decaf::util::Date::operator= ( const Date & value )`

Assigns the value of one **Date** (p. 1633) object to another.

#### Parameters

<i>value</i>	The value to be copied into this <b>Date</b> (p. 1633) object.
--------------	--

#### Returns

reference to this object with the newly assigned value.

6.303.3.8 `virtual bool decaf::util::Date::operator== ( const Date & value ) const`  
[virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

6.303.3.9 `void decaf::util::Date::setTime ( long long milliseconds )`

Sets the underlying time.

#### Parameters

<i>milliseconds</i>	The underlying time value in milliseconds.
---------------------	--

### 6.303.3.10 `std::string decaf::util::Date::toString ( ) const`

Converts this **Date** (p. 1633) object to a String of the form:

dow mon dd hh:mm:ss zzz yyyy

where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- mon is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- dd is the day of the month (01 through 31), as two decimal digits.
- hh is the hour of the day (00 through 23), as two decimal digits.
- mm is the minute within the hour (00 through 59), as two decimal digits.
- ss is the second within the minute (00 through 61, as two decimal digits).
- zzz is the time zone (and may reflect daylight saving time). Standard time zone abbreviations include those recognized by the method parse. If time zone information is not available, then zzz is empty - that is, it consists of no characters at all.
- yyyy is the year, as four decimal digits.

#### Returns

the String representation of the **Date** (p. 1633) object.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Date.h`

## 6.304 `decaf::internal::DecafRuntime` Class Reference

Handles APR initialization and termination.

```
#include <src/main/decaf/internal/DecafRuntime.h>
```

Inheritance diagram for `decaf::internal::DecafRuntime`:

#### Public Member Functions

- **DecafRuntime** ()  
*Initializes the APR Runtime for a library.*
- virtual **~DecafRuntime** ()  
*Terminates the APR Runtime for a library.*
- `apr_pool_t *` **getGlobalPool** () const  
*Grants access to the Global APR Pool instance that should be used when creating new threads.*

### 6.304.1 Detailed Description

Handles APR initialization and termination.

### 6.304.2 Constructor & Destructor Documentation

#### 6.304.2.1 decaf::internal::DecafRuntime::DecafRuntime ( )

Initializes the APR Runtime for a library.

#### 6.304.2.2 virtual decaf::internal::DecafRuntime::~~DecafRuntime ( ) [virtual]

Terminates the APR Runtime for a library.

### 6.304.3 Member Function Documentation

#### 6.304.3.1 apr\_pool\_t\* decaf::internal::DecafRuntime::getGlobalPool ( ) const

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**DecafRuntime.h**

## 6.305 activemq::threads::DedicatedTaskRunner Class Reference

```
#include <src/main/activemq/threads/DedicatedTaskRunner.h>
```

Inheritance diagram for activemq::threads::DedicatedTaskRunner:

### Public Member Functions

- **DedicatedTaskRunner** (Task \*task)
- virtual ~**DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)
 

*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()
 

*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()

Signal the **TaskRunner** (p. 3680) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3678) instance will be run until its `iterate` method has returned false indicating it is done.

## Protected Member Functions

- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

### 6.305.1 Constructor & Destructor Documentation

6.305.1.1 `activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner ( Task * task )`

6.305.1.2 `virtual activemq::threads::DedicatedTaskRunner::~~DedicatedTaskRunner ( )`  
[virtual]

### 6.305.2 Member Function Documentation

6.305.2.1 `virtual void activemq::threads::DedicatedTaskRunner::run ( )` [protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3265).

6.305.2.2 `virtual void activemq::threads::DedicatedTaskRunner::shutdown ( )` [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 3681).

6.305.2.3 `virtual void activemq::threads::DedicatedTaskRunner::shutdown ( unsigned int timeout )` [virtual]

Shutdown after a timeout, does not guarantee that the task's `iterate` method has completed and the thread halted.

#### Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implements **activemq::threads::TaskRunner** (p. 3681).

6.305.2.4 virtual void activemq::threads::DedicatedTaskRunner::wakeup ( ) [virtual]

Signal the **TaskRunner** (p. 3680) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3678) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 3681).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**DedicatedTaskRunner.h**

## 6.306 activemq::core::policies::DefaultPrefetchPolicy Class Reference

```
#include <src/main/activemq/core/policies/DefaultPrefetchPolicy.h>
```

Inheritance diagram for activemq::core::policies::DefaultPrefetchPolicy:

### Public Member Functions

- **DefaultPrefetchPolicy** ( )
- virtual ~**DefaultPrefetchPolicy** ( )
- virtual void **setDurableTopicPrefetch** (int value)  
*Sets the amount of prefetched messages for a Durable Topic.*
- virtual int **getDurableTopicPrefetch** ( ) const  
*Gets the amount of messages to prefetch for a Durable Topic.*
- virtual void **setQueuePrefetch** (int value)  
*Sets the amount of prefetched messages for a Queue.*
- virtual int **getQueuePrefetch** ( ) const  
*Gets the amount of messages to prefetch for a Queue.*
- virtual void **setQueueBrowserPrefetch** (int value)  
*Sets the amount of prefetched messages for a Queue Browser.*
- virtual int **getQueueBrowserPrefetch** ( ) const  
*Gets the amount of messages to prefetch for a Queue Browser.*
- virtual void **setTopicPrefetch** (int value)  
*Sets the amount of prefetched messages for a Topic.*
- virtual int **getTopicPrefetch** ( ) const  
*Gets the amount of messages to prefetch for a Topic.*
- virtual int **getMaxPrefetchLimit** (int value) const  
*Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.*
- virtual **PrefetchPolicy** \* **clone** ( ) const  
*Clone the Policy and return a new pointer to that clone.*

## Static Public Attributes

- static int **MAX\_PREFETCH\_SIZE**
- static int **DEFAULT\_DURABLE\_TOPIC\_PREFETCH**
- static int **DEFAULT\_QUEUE\_PREFETCH**
- static int **DEFAULT\_QUEUE\_BROWSER\_PREFETCH**
- static int **DEFAULT\_TOPIC\_PREFETCH**

## 6.306.1 Constructor & Destructor Documentation

6.306.1.1 `activemq::core::policies::DefaultPrefetchPolicy::DefaultPrefetchPolicy ( )`

6.306.1.2 `virtual activemq::core::policies::DefaultPrefetchPolicy::~~DefaultPrefetchPolicy ( )`  
[virtual]

## 6.306.2 Member Function Documentation

6.306.2.1 `virtual PrefetchPolicy* activemq::core::policies::DefaultPrefetchPolicy::clone ( )`  
`const` [virtual]

Clone the Policy and return a new pointer to that clone.

### Returns

pointer to a new **PrefetchPolicy** (p. 2924) instance that is a clone of this one.

Implements **activemq::core::PrefetchPolicy** (p. 2926).

6.306.2.2 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getDurableTopicPrefetch ( )const` [inline, virtual]

Gets the amount of messages to prefetch for a Durable Topic.

### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2926).

6.306.2.3 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getMaxPrefetchLimit ( int value )const` [inline, virtual]

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

### Returns

the allowable value for a prefetch limit, either requested or the max.

Implements **activemq::core::PrefetchPolicy** (p. 2927).

6.306.2.4 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueueBrowserPrefetch ( ) const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue Browser.

#### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2927).

6.306.2.5 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getQueuePrefetch ( ) const [inline, virtual]`

Gets the amount of messages to prefetch for a Queue.

#### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2927).

6.306.2.6 `virtual int activemq::core::policies::DefaultPrefetchPolicy::getTopicPrefetch ( ) const [inline, virtual]`

Gets the amount of messages to prefetch for a Topic.

#### Returns

value of the number of messages to prefetch.

Implements **activemq::core::PrefetchPolicy** (p. 2927).

6.306.2.7 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setDurableTopicPrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Durable Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2928).

6.306.2.8 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueueBrowserPrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Queue Browser.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2928).

6.306.2.9 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setQueuePrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Queue.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2928).

6.306.2.10 `virtual void activemq::core::policies::DefaultPrefetchPolicy::setTopicPrefetch ( int value ) [inline, virtual]`

Sets the amount of prefetched messages for a Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implements **activemq::core::PrefetchPolicy** (p. 2928).

### 6.306.3 Field Documentation

6.306.3.1 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_DURABLE_TOPIC_PREFETCH [static]`

6.306.3.2 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_BROWSER_PREFETCH [static]`

6.306.3.3 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_QUEUE_PREFETCH [static]`

6.306.3.4 `int activemq::core::policies::DefaultPrefetchPolicy::DEFAULT_TOPIC_PREFETCH [static]`



6.306.3.5 int **activemq::core::policies::DefaultPrefetchPolicy::MAX\_PREFETCH\_SIZE** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/policies/**DefaultPrefetchPolicy.h**

## 6.307 activemq::core::policies::DefaultRedeliveryPolicy Class Reference

```
#include <src/main/activemq/core/policies/DefaultRedeliveryPolicy.h>
```

Inheritance diagram for **activemq::core::policies::DefaultRedeliveryPolicy**:

### Public Member Functions

- **DefaultRedeliveryPolicy** ()
- virtual **~DefaultRedeliveryPolicy** ()
- virtual double **getBackOffMultiplier** () const
- virtual void **setBackOffMultiplier** (double value)  
*Sets the Back-Off Multiplier for Message Redelivery.*
- virtual short **getCollisionAvoidancePercent** () const
- virtual void **setCollisionAvoidancePercent** (short value)
- virtual long long **getInitialRedeliveryDelay** () const  
*Gets the initial time that redelivery of messages is delayed.*
- virtual void **setInitialRedeliveryDelay** (long long value)  
*Sets the initial time that redelivery will be delayed.*
- virtual int **getMaximumRedeliveries** () const  
*Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.*
- virtual void **setMaximumRedeliveries** (int value)  
*Sets the Maximum allowable redeliveries for a Message.*
- virtual bool **isUseCollisionAvoidance** () const
- virtual void **setUseCollisionAvoidance** (bool value)
- virtual bool **isUseExponentialBackOff** () const
- virtual void **setUseExponentialBackOff** (bool value)
- virtual long long **getRedeliveryDelay** (long long previousDelay)  
*Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.*
- virtual **RedeliveryPolicy \* clone** () const  
*Create a copy of this Policy and return it.*

### 6.307.1 Constructor & Destructor Documentation

6.307.1.1 `activemq::core::policies::DefaultRedeliveryPolicy::DefaultRedeliveryPolicy ( )`

6.307.1.2 `virtual activemq::core::policies::DefaultRedeliveryPolicy::~~DefaultRedeliveryPolicy ( ) [virtual]`

### 6.307.2 Member Function Documentation

6.307.2.1 `virtual RedeliveryPolicy* activemq::core::policies::DefaultRedeliveryPolicy::clone ( ) const [virtual]`

Create a copy of this Policy and return it.

#### Returns

pointer to a new **RedeliveryPolicy** (p. 3121) that is a copy of this one.

Implements **activemq::core::RedeliveryPolicy** (p. 3123).

6.307.2.2 `virtual double activemq::core::policies::DefaultRedeliveryPolicy::getBackOffMultiplier ( ) const [inline, virtual]`

#### Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 3123).

6.307.2.3 `virtual short activemq::core::policies::DefaultRedeliveryPolicy::getCollisionAvoidancePercent ( ) const [virtual]`

#### Returns

the currently set Collision Avoidance percentage.

Implements **activemq::core::RedeliveryPolicy** (p. 3124).

6.307.2.4 `virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getInitialRedeliveryDelay ( ) const [inline, virtual]`

Gets the initial time that redelivery of messages is delayed.

#### Returns

the time in milliseconds that redelivery is delayed initially.

Implements **activemq::core::RedeliveryPolicy** (p. 3124).

6.307.2.5 `virtual int activemq::core::policies::DefaultRedeliveryPolicy::getMaximumRedeliveries ( ) const [inline, virtual]`

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

#### Returns

maximum allowed redeliveries for a message.

Implements **activemq::core::RedeliveryPolicy** (p. 3124).

6.307.2.6 `virtual long long activemq::core::policies::DefaultRedeliveryPolicy::getRedeliveryDelay ( long long previousDelay ) [virtual]`

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

#### Parameters

<i>previousDelay</i>	The last delay that was used between message redeliveries.
----------------------	--

#### Returns

the new delay to use before attempting another redelivery.

Implements **activemq::core::RedeliveryPolicy** (p. 3124).

6.307.2.7 `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseCollisionAvoidance ( ) const [inline, virtual]`

#### Returns

whether or not collision avoidance is enabled for this Policy.

Implements **activemq::core::RedeliveryPolicy** (p. 3125).

6.307.2.8 `virtual bool activemq::core::policies::DefaultRedeliveryPolicy::isUseExponentialBackOff ( ) const [inline, virtual]`

#### Returns

whether or not the exponential back off option is enabled.

Implements **activemq::core::RedeliveryPolicy** (p. 3125).

6.307.2.9 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setBackOffMultiplier (double value) [inline, virtual]`

Sets the Back-Off Multiplier for Message Redelivery.

#### Parameters

<i>value</i>	The new value for the back-off multiplier.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 3125).

6.307.2.10 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setCollisionAvoidancePercent (short value) [virtual]`

#### Parameters

<i>value</i>	The collision avoidance percentage setting.
--------------	---

Implements **activemq::core::RedeliveryPolicy** (p. 3125).

6.307.2.11 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setInitialRedeliveryDelay (long long value) [inline, virtual]`

Sets the initial time that redelivery will be delayed.

#### Parameters

<i>value</i>	Time in Milliseconds to wait before starting redelivery.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 3125).

6.307.2.12 `virtual void activemq::core::policies::DefaultRedeliveryPolicy::setMaximumRedeliveries (int maximumRedeliveries) [inline, virtual]`

Sets the Maximum allowable redeliveries for a Message.

#### Parameters

<i>maximum-Redeliveries</i>	The maximum number of times that a message will be redelivered.
-----------------------------	---

Implements **activemq::core::RedeliveryPolicy** (p. 3126).

6.307.2.13 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseCollisionAvoidance ( bool *value* ) [inline, virtual]

#### Parameters

<i>value</i>	Enable or Disable collision avoidance for this Policy.
--------------	--

Implements **activemq::core::RedeliveryPolicy** (p. 3126).

6.307.2.14 virtual void activemq::core::policies::DefaultRedeliveryPolicy::setUseExponentialBackOff ( bool *value* ) [inline, virtual]

#### Parameters

<i>value</i>	Enable or Disable the exponential back off multiplier option.
--------------	---

Implements **activemq::core::RedeliveryPolicy** (p. 3126).

The documentation for this class was generated from the following file:

- src/main/activemq/core/policies/**DefaultRedeliveryPolicy.h**

## 6.308 decaf::internal::net::DefaultServerSocketFactory Class Reference

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

```
#include <src/main/decaf/internal/net/DefaultServerSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::DefaultServerSocketFactory:

### Public Member Functions

- **DefaultServerSocketFactory** ()
- virtual ~**DefaultServerSocketFactory** ()
- virtual **decaf::net::ServerSocket** \* **createServerSocket** ()

Create a new **ServerSocket** (p. 3292) that is unbound.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

#### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

#### Exceptions

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
------	--

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will use the specified connection backlog setting.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog, const **decaf::net::InetAddress \*address**)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL then the **ServerSocket** (p. 3292) will listen on all interfaces.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.
address	The address of the interface on the local machine to bind to.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

### 6.308.1 Detailed Description

Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.

**Since**

1.0

**6.308.2 Constructor & Destructor Documentation**

6.308.2.1 decaf::internal::net::DefaultServerSocketFactory::DefaultServerSocketFactory ( )

6.308.2.2 virtual decaf::internal::net::DefaultServerSocketFactory::~DefaultServerSocketFactory ( ) [virtual]

**6.308.3 Member Function Documentation**

6.308.3.1 virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( ) [virtual]

Create a new **ServerSocket** (p. 3292) that is unbound.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3302).

6.308.3.2 virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( int *port*, int *backlog*, const decaf::net::InetAddress \* *address* ) [virtual]

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3292) will listen on all interfaces.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3303).

6.308.3.3 **virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( int *port*, int *backlog* )** [virtual]

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

The **ServerSocket** (p. 3292) will use the specified connection backlog setting.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3304).

6.308.3.4 **virtual decaf::net::ServerSocket\* decaf::internal::net::DefaultServerSocketFactory::createServerSocket ( int *port* )** [virtual]

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
-------------	--

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.



**Exceptions**

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3303).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**DefaultServerSocketFactory.h**

**6.309 decaf::internal::net::DefaultSocketFactory Class Reference**

SocketFactory implementation that is used to create Sockets.

```
#include <src/main/decaf/internal/net/DefaultSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::DefaultSocketFactory:

**Public Member Functions**

- **DefaultSocketFactory** ()
- virtual **~DefaultSocketFactory** ()
- virtual **decaf::net::Socket \* createSocket** () throw ( decaf::io::IOException )

*Creates an unconnected **Socket** (p. 3445) object.*

**Returns**

*a new **Socket** (p. 3445) object, caller must free this object when done.*

**Exceptions**

<i>IOException</i>	if the <b>Socket</b> (p. 3445) cannot be created.
--------------------	---

- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

**Parameters**

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

**Returns**

*a new **Socket** (p. 3445) object, caller must free this object when done.*

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host,

int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

*The **Socket** (p. 3445) will be bound to the specified local address and port.*

#### Parameters

host	<i>The host to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>
ifAddress	<i>The address on the local machine to bind the <b>Socket</b> (p. 3445) to.</i>
localPort	<i>The local port to bind the <b>Socket</b> (p. 3445) to.</i>

#### Returns

*a new **Socket** (p. 3445) object, caller must free this object when done.*

#### Exceptions

IOException	<i>if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.</i>
<b>UnknownHostException</b> (p. 3841)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket** \* **createSocket** (const std::string &name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

#### Parameters

host	<i>The host name or IP address to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>

#### Returns

*a new **Socket** (p. 3445) object, caller must free this object when done.*

#### Exceptions

IOException	<i>if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.</i>
<b>UnknownHostException</b> (p. 3841)	<i>if the host name is not known.</i>

- virtual **decaf::net::Socket** \* **createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

#### Parameters

host	<i>The host name or IP address to connect the socket to.</i>
port	<i>The port on the remote host to connect to.</i>
ifAddress	<i>The address on the local machine to bind the <b>Socket</b> (p. 3445) to.</i>
localPort	<i>The local port to bind the <b>Socket</b> (p. 3445) to.</i>

#### Returns

*a new **Socket** (p. 3445) object, caller must free this object when done.*

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

**6.309.1 Detailed Description**

SocketFactory implementation that is used to create Sockets.

**Since**

1.0

**6.309.2 Constructor & Destructor Documentation**

6.309.2.1 decaf::internal::net::DefaultSocketFactory::DefaultSocketFactory ( )

6.309.2.2 virtual decaf::internal::net::DefaultSocketFactory::~~DefaultSocketFactory ( )  
[virtual]

**6.309.3 Member Function Documentation**

6.309.3.1 virtual decaf::net::Socket\* decaf::internal::net::DefaultSocketFactory::createSocket  
( ) throw ( decaf::io::IOException ) [virtual]

Creates an unconnected **Socket** (p. 3445) object.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

IOException	if the <b>Socket</b> (p. 3445) cannot be created.
-------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 3468).

6.309.3.2 virtual decaf::net::Socket\* decaf::internal::net::DefaultSocketFactory::createSocket  
( const std::string & name, int port, const decaf::net::InetAddress  
\* ifAddress, int localPort ) throw ( decaf::io::IOException,  
decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

**Parameters**

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3469).

```
6.309.3.3  virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket
            ( const std::string & name, int port ) throw ( decaf::io::IOException,
            decaf::net::UnknownHostException ) [virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

**Parameters**

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3470).

```
6.309.3.4 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket
( const decaf::net::InetAddress * host, int port, const de-
caf::net::InetAddress * ifAddress, int localPort ) throw (
decaf::io::IOException, decaf::net::UnknownHostException )
[virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

The **Socket** (p. 3445) will be bound to the specified local address and port.

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3470).

```
6.309.3.5 virtual decaf::net::Socket* decaf::internal::net::DefaultSocketFactory::createSocket
( const decaf::net::InetAddress * host, int port ) throw ( de-
caf::io::IOException, decaf::net::UnknownHostException )
[virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
--------------------	---

<b><i>UnknownHostException</i></b> (p. 3841)	if the host name is not known.
---	--------------------------------

Implements **decaf::net::SocketFactory** (p. 3469).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/DefaultSocketFactory.h`

## 6.310 decaf::internal::net::ssl::DefaultSSLContext Class Reference

Default SSLContext manager for the Decaf library.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLContext.h>
```

### Public Member Functions

- virtual `~DefaultSSLContext()`

### Static Public Member Functions

- static `decaf::net::ssl::SSLContext * getContext()`

### Protected Member Functions

- `DefaultSSLContext()`

#### 6.310.1 Detailed Description

Default SSLContext manager for the Decaf library.

If the user doesn't supply or specify the SSLContext that they wish to use then we load the Decaf library's default SSLContext using whatever SSL provider is enabled and preferred.

#### Since

1.0

#### 6.310.2 Constructor & Destructor Documentation

6.310.2.1 `decaf::internal::net::ssl::DefaultSSLContext::DefaultSSLContext()`  
 [protected]

## 6.311 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference 1005

6.310.2.2 virtual decaf::internal::net::ssl::DefaultSSLContext::~~DefaultSSLContext ( )  
[virtual]

### 6.310.3 Member Function Documentation

6.310.3.1 static decaf::net::ssl::SSLContext\* decaf::internal::net::ssl::DefaultSSLContext::getContext ( )  
[static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/DefaultSSLContext.h

## 6.311 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLServerSocketFactory:

### Public Member Functions

- **DefaultSSLServerSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLServerSocketFactory** ( )
- virtual decaf::net::ServerSocket \* **createServerSocket** ( )

Create a new **ServerSocket** (p. 3292) that is unbound.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

#### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

#### Exceptions

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual decaf::net::ServerSocket \* **createServerSocket** (int port)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

#### Parameters

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
------	--

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will use the specified connection backlog setting.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog, const **decaf::net::InetAddress** \*address)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3292) will listen on all interfaces.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.
address	The address of the interface on the local machine to bind to.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **std::vector< std::string > getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3506)

- virtual **std::vector< std::string > getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those



## 6.311 decaf::internal::net::ssl::DefaultSSLServerSocketFactory Class Reference

defaults. Such cipher suites are useful in specialized applications.

### Returns

an STL vector containing the list of supported cipher suites.

### See also

**getDefaultCipherSuites()** (p. 3505)

### 6.311.1 Detailed Description

Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

### Since

1.0

### 6.311.2 Constructor & Destructor Documentation

6.311.2.1 decaf::internal::net::ssl::DefaultSSLServerSocketFactory::DefaultSSLServerSocketFactory  
( const std::string & errorMessage )

6.311.2.2 virtual decaf::internal::net::ssl::DefaultSSLServerSocketFactory::~DefaultSSLServerSocketFactory  
( ) [virtual]

### 6.311.3 Member Function Documentation

6.311.3.1 virtual decaf::net::ServerSocket\* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( )  
[virtual]

Create a new **ServerSocket** (p. 3292) that is unbound.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3302).

6.311.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( int port ) [virtual]`

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
-------------	--

#### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3303).

6.311.3.3 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket ( int port, int backlog, const decaf::net::InetAddress * address ) [virtual]`

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

The **ServerSocket** (p. 3292) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3292) will listen on all interfaces.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

#### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3303).

## 6.311 **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** Class Reference 1009

6.311.3.4 virtual **decaf::net::ServerSocket**\* **decaf::internal::net::ssl::DefaultSSLServerSocketFactory::createServerSocket** ( int *port*, int *backlog* ) [virtual]

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.  
The **ServerSocket** (p. 3292) will use the specified connection backlog setting.

### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.

### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3304).

6.311.3.5 virtual std::vector<std::string> **decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getDefaultCipherSuites** ( ) [virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

### Returns

an STL vector containing the list of cipher suites enabled by default.

### See also

**getSupportedCipherSuites()** (p. 3506)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3505).

6.311.3.6 virtual std::vector<std::string> **decaf::internal::net::ssl::DefaultSSLServerSocketFactory::getSupportedCipherSuites** ( ) [virtual]

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

### Returns

an STL vector containing the list of supported cipher suites.

### See also

**getDefaultCipherSuites()** (p. 3505)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3506).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/**DefaultSSLServerSocketFactory.h**

## 6.312 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

```
#include <src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h>
```

Inheritance diagram for decaf::internal::net::ssl::DefaultSSLSocketFactory:

### Public Member Functions

- **DefaultSSLSocketFactory** (const std::string &errorMessage)
- virtual ~**DefaultSSLSocketFactory** ()
- virtual **decaf::net::Socket** \* **createSocket** () throw ( decaf::io::IOException )

*Creates an unconnected **Socket** (p. 3445) object.*

#### Returns

*a new **Socket** (p. 3445) object, caller must free this object when done.*

#### Exceptions

IOException	if the <b>Socket</b> (p. 3445) cannot be created.
-------------	---

- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

## 6.312 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference 1671

### Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.

### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

### Exceptions

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

The **Socket** (p. 3445) will be bound to the specified local address and port.

### Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
localPort	The local port to bind the <b>Socket</b> (p. 3445) to.

### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

### Exceptions

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

### Parameters

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

### Exceptions

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException,

decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

**Parameters**

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
localPort	The local port to bind the <b>Socket</b> (p. 3445) to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3517)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites()** (p. 3517)

- virtual decaf::net::Socket \* **createSocket** (decaf::net::Socket \*socket, std::string host, int port, bool autoClose)

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters**

socket	The existing socket to layer over.
host	The server host the original <b>Socket</b> (p. 3445) is connected to.
port	The server port the original <b>Socket</b> (p. 3445) is connected to.
autoClose	Should the layered over <b>Socket</b> (p. 3445) be closed when the topmost socket is closed.

**Returns**

a new **Socket** (p. 3445) instance that wraps the given **Socket** (p. 3445).

**Exceptions**

IOException	if an I/O exception occurs while performing this operation.
<b>UnknownHostException</b> (p. 3841)	if the host is unknown.

**6.312.1 Detailed Description**

Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.

**Since**

1.0

**6.312.2 Constructor & Destructor Documentation**

6.312.2.1 `decaf::internal::net::ssl::DefaultSSLSocketFactory::DefaultSSLSocketFactory ( const std::string & errorMessage )`

6.312.2.2 `virtual decaf::internal::net::ssl::DefaultSSLSocketFactory::~~DefaultSSLSocketFactory ( ) [virtual]`

**6.312.3 Member Function Documentation**

6.312.3.1 `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( ) throw ( decaf::io::IOException ) [virtual]`

Creates an unconnected **Socket** (p. 3445) object.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

IOException	if the <b>Socket</b> (p. 3445) cannot be created.
-------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 3468).

6.312.3.2 `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( decaf::net::Socket * socket, std::string host, int port, bool autoClose )`  
[virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

#### Parameters

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original <b>Socket</b> (p. 3445) is connected to.
<i>port</i>	The server port the original <b>Socket</b> (p. 3445) is connected to.
<i>autoClose</i>	Should the layered over <b>Socket</b> (p. 3445) be closed when the topmost socket is closed.

#### Returns

a new **Socket** (p. 3445) instance that wraps the given **Socket** (p. 3445).

#### Exceptions

<i>IOException</i>	if an I/O exception occurs while performing this operation.
<b>UnknownHostException</b> (p. 3841)	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3516).

6.312.3.3 `virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket ( const decaf::net::InetAddress * host, int port ) throw ( decaf::io::IOException, decaf::net::UnknownHostException )`  
[virtual]

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
--------------------	---



## 6.312 decaf::internal::net::ssl::DefaultSSLSocketFactory Class Reference 1675

<b><i>UnknownHostException</i></b> (p. 3841)	if the host name is not known.
---	--------------------------------

Implements **decaf::net::SocketFactory** (p. 3469).

6.312.3.4 virtual **decaf::net::Socket\*** **decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket**  
( const std::string & *name*, int *port* ) throw ( **decaf::io::IOException**,  
**decaf::net::UnknownHostException** ) [virtual]

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b><i>UnknownHostException</i></b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3470).

6.312.3.5 virtual **decaf::net::Socket\*** **decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket**  
( const std::string & *name*, int *port*, const **decaf::net::InetAddress**  
\* *ifAddress*, int *localPort* ) throw ( **decaf::io::IOException**,  
**decaf::net::UnknownHostException** ) [virtual]

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3469).

```
6.312.3.6 virtual decaf::net::Socket* decaf::internal::net::ssl::DefaultSSLSocketFactory::createSocket
( const decaf::net::InetAddress * host, int port, const de-
caf::net::InetAddress * ifAddress, int localPort ) throw (
decaf::io::IOException, decaf::net::UnknownHostException )
[virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

The **Socket** (p. 3445) will be bound to the specified local address and port.

**Parameters**

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3470).

```
6.312.3.7 virtual std::vector<std::string> de-
caf::internal::net::ssl::DefaultSSLSocketFactory::getDefaultCipherSuites ( )
[virtual]
```

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3517)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3517).

```
6.312.3.8 virtual std::vector<std::string> de-
caf::internal::net::ssl::DefaultSSLSocketFactory::getSupportedCipherSuites ( )
[virtual]
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites()** (p. 3517)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3517).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/**DefaultSSLSocketFactory.h**

**6.313 activemq::transport::DefaultTransportListener Class Reference**

```
#include <src/main/activemq/transport/DefaultTransportListener.h>
```

Inheritance diagram for activemq::transport::DefaultTransportListener:

**Public Member Functions**

- virtual **~DefaultTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command AMQCPP\_ - UNUSED)

*Event handler for the receipt of a command.*

- virtual void **onException** (const **decaf::lang::Exception** &ex *AMQCPP\_UNUSED*)

*Event handler for an exception from a command transport.*

- virtual void **transportInterrupted** ()

*The transport has suffered an interruption from which it hopes to recover.*

- virtual void **transportResumed** ()

*The transport has resumed after an interruption.*

### 6.313.1 Constructor & Destructor Documentation

6.313.1.1 virtual **activemq::transport::DefaultTransportListener::~~DefaultTransportListener** ( )  
[inline, virtual]

### 6.313.2 Member Function Documentation

6.313.2.1 virtual void **activemq::transport::DefaultTransportListener::onCommand** ( const **Pointer< Command >** &command *AMQCPP\_UNUSED* ) [inline, virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3819) deletes the command upon receipt.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

6.313.2.2 virtual void **activemq::transport::DefaultTransportListener::onException** ( const **decaf::lang::Exception** &ex *AMQCPP\_UNUSED* ) [inline, virtual]

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception.
-----------	----------------

6.313.2.3 virtual void **activemq::transport::DefaultTransportListener::transportInterrupted** ( )  
[inline, virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3837).

6.313.2.4 virtual void activemq::transport::DefaultTransportListener::transportResumed ( )  
[inline, virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3837).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**DefaultTransportListener.h**

## 6.314 decaf::util::zip::Deflater Class Reference

This class compresses data using the *DEFLATE* algorithm (see [specification](#)).

```
#include <src/main/decaf/util/zip/Deflater.h>
```

### Public Member Functions

- **Deflater** (int level, bool nowrap=false)  
*Creates a new compressor using the specified compression level.*
- **Deflater** ()  
*Creates a new compressor with the default compression level.*
- virtual ~**Deflater** ()
- void **setInput** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setInput** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Sets input data for compression.*
- void **setDictionary** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*
- void **setDictionary** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Sets preset dictionary for compression.*

- void **setStrategy** (int strategy) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Sets the compression strategy to the specified value.*
- void **setLevel** (int level) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Sets the compression level to the specified value.*
- bool **needsInput** () const
- void **finish** ()  
*When called, indicates that compression should end with the current contents of the input buffer.*
- bool **finished** () const
- int **deflate** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Fills specified buffer with compressed data.*
- int **deflate** (std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )  
*Fills specified buffer with compressed data.*
- int **deflate** (std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )  
*Fills specified buffer with compressed data.*
- long long **getAdler** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesRead** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesWritten** () const throw ( decaf::lang::exceptions::IllegalStateException )
- void **reset** () throw ( decaf::lang::exceptions::IllegalStateException )  
*Resets deflater so that a new set of input data can be processed.*
- void **end** ()  
*Closes the compressor and discards any unprocessed input.*

### Static Public Attributes

- static const int **BEST\_SPEED**  
*Compression level for fastest compression.*
- static const int **BEST\_COMPRESSION**  
*Compression level for best compression.*
- static const int **DEFAULT\_COMPRESSION**  
*Default compression level.*
- static const int **DEFLATED**  
*Compression method for the deflate algorithm (the only one currently supported).*
- static const int **NO\_COMPRESSION**  
*Compression level for no compression.*

- static const int **FILTERED**  
*Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.*
- static const int **HUFFMAN\_ONLY**  
*Compression strategy for Huffman coding only.*
- static const int **DEFAULT\_STRATEGY**  
*Default compression strategy.*

### 6.314.1 Detailed Description

This class compresses data using the *DEFLATE* algorithm (see *specification*).

Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **DeflaterOutputStream** (p. 1682) and its descendants.

The typical usage of a **Deflater** (p. 1672) instance outside this package consists of a specific call to one of its constructors before being passed to an instance of **DeflaterOutputStream** (p. 1682).

#### See also

**DeflaterOutputStream** (p. 1682)

**Inflater** (p. 1985)

#### Since

1.0

### 6.314.2 Constructor & Destructor Documentation

#### 6.314.2.1 decaf::util::zip::Deflater::Deflater ( int *level*, bool *nowrap* = false )

Creates a new compressor using the specified compression level.

If 'nowrap' is true then the ZLIB header and checksum fields will not be used in order to support the compression format used in both GZIP and PKZIP.

#### Parameters

<i>level</i>	The compression level to use (0-9).
<i>nowrap</i>	If true uses GZip compatible compression (defaults to false).

#### 6.314.2.2 decaf::util::zip::Deflater::Deflater ( )

Creates a new compressor with the default compression level.

Compressed data will be generated in ZLIB format.

6.314.2.3 virtual decaf::util::zip::Deflater::~Deflater ( ) [virtual]

### 6.314.3 Member Function Documentation

6.314.3.1 int decaf::util::zip::Deflater::deflate ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1677) should be called in order to determine if more input data is required.

#### Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>size</i>	The size of the passed buffer.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

#### Returns

the actual number of bytes of compressed data.

#### Exceptions

<i>NullPointerException</i>	if <i>buffer</i> is NULL.
<i>IndexOutOfBoundsException</i>	if the <i>offset</i> + <i>length</i> > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.314.3.2 int decaf::util::zip::Deflater::deflate ( std::vector< unsigned char > & *buffer*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1677) should be called in order to determine if more input data is required.

#### Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.



**Returns**

the actual number of bytes of compressed data.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.314.3.3 `int decaf::util::zip::Deflater::deflate ( std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Fills specified buffer with compressed data.

Returns actual number of bytes of compressed data. A return value of 0 indicates that **needsInput()** (p. 1677) should be called in order to determine if more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to write the compressed data to.
---------------	---

**Returns**

the actual number of bytes of compressed data.

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.314.3.4 `void decaf::util::zip::Deflater::end ( )`

Closes the compressor and discards any unprocessed input.

This method should be called when the compressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Deflater** (p. 1672) object is undefined.

6.314.3.5 `void decaf::util::zip::Deflater::finish ( )`

When called, indicates that compression should end with the current contents of the input buffer.

6.314.3.6 `bool decaf::util::zip::Deflater::finished ( ) const`

**Returns**

true if the end of the compressed data output stream has been reached.

6.314.3.7 `long long decaf::util::zip::Deflater::getAdler ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the ADLER-32 value of the uncompressed data.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.314.3.8 `long long decaf::util::zip::Deflater::getBytesRead ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of uncompressed bytes input so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.314.3.9 `long long decaf::util::zip::Deflater::getBytesWritten ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of compressed bytes output so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.314.3.10 `bool decaf::util::zip::Deflater::needsInput ( ) const`

#### Returns

true if the input data buffer is empty and **setInput()** (p. 1680) should be called in order to provide more input

6.314.3.11 `void decaf::util::zip::Deflater::reset ( ) throw ( decaf::lang::exceptions::IllegalStateException )`

Resets deflater so that a new set of input data can be processed.

Keeps current compression level and strategy settings.

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.314.3.12 void decaf::util::zip::Deflater::setDictionary ( const std::vector< unsigned char > & *buffer*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1990), **Inflater.getAdler()** (p. 1988) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters**

<i>buffer</i>	The buffer containing the preset dictionary.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.314.3.13 void decaf::util::zip::Deflater::setDictionary ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1990), **Inflater.getAdler()** (p. 1988) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters**

<i>buffer</i>	The buffer containing the preset dictionary.
<i>size</i>	The size of the passed dictionary buffer.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.314.3.14 `void decaf::util::zip::Deflater::setDictionary ( const std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Sets preset dictionary for compression.

A preset dictionary is used when the history buffer can be predetermined. When the data is later uncompressed with **Inflater.inflate()** (p. 1990), **Inflater.getAdler()** (p. 1988) can be called in order to get the Adler-32 value of the dictionary required for decompression.

**Parameters**

<i>buffer</i>	The buffer containing the preset dictionary.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.314.3.15 `void decaf::util::zip::Deflater::setInput ( const std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1677) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for compression.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.314.3.16 `void decaf::util::zip::Deflater::setInput ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )`

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1677) returns true indicating that more

input data is required.

#### Parameters

<i>buffer</i>	The Buffer to read in for compression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.314.3.17 void decaf::util::zip::Deflater::setInput ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for compression.

This should be called whenever **needsInput()** (p. 1677) returns true indicating that more input data is required.

#### Parameters

<i>buffer</i>	The Buffer to read in for compression.
<i>size</i>	The size in bytes of the buffer passed.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.314.3.18 void decaf::util::zip::Deflater::setLevel ( int *level* ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Sets the compression level to the specified value.

#### Parameters

<i>level</i>	The new Compression level to use.
--------------	-----------------------------------

**Exceptions**

<i>IllegalArgumentException</i>	if the level value is invalid.
<i>IllegalStateException</i>	if in the end state.

**6.314.3.19** `void decaf::util::zip::Deflater::setStrategy ( int strategy ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Sets the compression strategy to the specified value.

**Parameters**

<i>strategy</i>	The new Compression strategy to use.
-----------------	--------------------------------------

**Exceptions**

<i>IllegalArgumentException</i>	if the strategy value is invalid.
<i>IllegalStateException</i>	if in the end state.

**6.314.4 Field Documentation**

**6.314.4.1** `const int decaf::util::zip::Deflater::BEST_COMPRESSION [static]`

Compression level for best compression.

**6.314.4.2** `const int decaf::util::zip::Deflater::BEST_SPEED [static]`

Compression level for fastest compression.

**6.314.4.3** `const int decaf::util::zip::Deflater::DEFAULT_COMPRESSION [static]`

Default compression level.

**6.314.4.4** `const int decaf::util::zip::Deflater::DEFAULT_STRATEGY [static]`

Default compression strategy.

**6.314.4.5** `const int decaf::util::zip::Deflater::DEFLATED [static]`

Compression method for the deflate algorithm (the only one currently supported).

## 6.314.4.6 const int decaf::util::zip::Deflater::FILTERED [static]

Compression strategy best used for data consisting mostly of small values with a somewhat random distribution.

Forces more Huffman coding and less string matching.

## 6.314.4.7 const int decaf::util::zip::Deflater::HUFFMAN\_ONLY [static]

Compression strategy for Huffman coding only.

## 6.314.4.8 const int decaf::util::zip::Deflater::NO\_COMPRESSION [static]

Compression level for no compression.

The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Deflater.h**

## 6.315 decaf::util::zip::DeflaterOutputStream Class Reference

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

```
#include <src/main/decaf/util/zip/DeflaterOutputStream.h>
```

Inheritance diagram for decaf::util::zip::DeflaterOutputStream:

## Public Member Functions

- **DeflaterOutputStream** (decaf::io::OutputStream \*outputStream, bool own=false)  
*Creates a new DeflateOutputStream with a Default **Deflater** (p. 1672) and buffer size.*
- **DeflaterOutputStream** (decaf::io::OutputStream \*outputStream, Deflater \*deflater, bool own=false)  
*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1672) and a default buffer size.*
- **DeflaterOutputStream** (decaf::io::OutputStream \*outputStream, Deflater \*deflater, int bufferSize, bool own=false)  
*Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1672) and specified buffer size.*
- virtual ~**DeflaterOutputStream** ()
- virtual void **finish** () throw ( decaf::io::IOException )  
*Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.*

*The object is generally no longer usable after calling close.*

#### **Exceptions**

<b>IOException</b> (p. 2103)	if an error occurs while closing.
------------------------------	-----------------------------------

*The default implementation of this method does nothing.*

*The close method of **FilterOutputStream** (p. 1861) calls its flush method, and then calls the close method of its underlying output stream.*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **deflate** () throw ( decaf::io::IOException )

*Writes a buffers worth of compressed data to the wrapped OutputStream.*

### Protected Attributes

- **Deflater** \* **deflater**

*The **Deflater** (p. 1672) for this stream.*

- std::vector< unsigned char > **buf**

*The Buffer to use for.*

- bool **ownDeflater**
- bool **isDone**

### Static Protected Attributes

- static const std::size\_t **DEFAULT\_BUFFER\_SIZE**

#### 6.315.1 Detailed Description

Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.

#### Since

1.0



## 6.315.2 Constructor & Destructor Documentation

6.315.2.1 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream ( decaf::io::OutputStream * outputStream, bool own = false )`

Creates a new DeflateOutputStream with a Default **Deflater** (p. 1672) and buffer size.

### Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).

6.315.2.2 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream ( decaf::io::OutputStream * outputStream, Deflater * deflater, bool own = false )`

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1672) and a default buffer size.

When the user supplied a **Deflater** (p. 1672) instance the DeflaterOutputStream does not take ownership of the **Deflater** (p. 1672) pointer, the caller is still responsible for deleting the **Deflater** (p. 1672).

### Parameters

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>deflater</i>	The user supplied <b>Deflater</b> (p. 1672) to use for compression. (
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).

### Exceptions

<i>NullPointerException</i>	if the <b>Deflater</b> (p. 1672) given is NULL.
-----------------------------	---

6.315.2.3 `decaf::util::zip::DeflaterOutputStream::DeflaterOutputStream ( decaf::io::OutputStream * outputStream, Deflater * deflater, int bufferSize, bool own = false )`

Creates a new DeflateOutputStream with a user supplied **Deflater** (p. 1672) and specified buffer size.

When the user supplied a **Deflater** (p. 1672) instance the DeflaterOutputStream does not take ownership of the **Deflater** (p. 1672) pointer, the caller is still responsible for deleting the **Deflater** (p. 1672).

**Parameters**

<i>output-Stream</i>	The OutputStream instance to wrap.
<i>deflater</i>	The user supplied <b>Deflater</b> (p. 1672) to use for compression.
<i>bufferSize</i>	The size of the input buffer.
<i>own</i>	Should this filter take ownership of the OutputStream pointer (default is false).

**Exceptions**

<i>NullPointerException</i>	if the <b>Deflater</b> (p. 1672) given is NULL.
<i>IllegalArgumentException</i>	if bufferSize is 0.

6.315.2.4 `virtual decaf::util::zip::DeflaterOutputStream::~~DeflaterOutputStream ( )`  
[virtual]

**6.315.3 Member Function Documentation**

6.315.3.1 `virtual void decaf::util::zip::DeflaterOutputStream::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

**Exceptions**

<b>IOException</b> (p. 2103)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1861) calls its flush method, and then calls the close method of its underlying output stream.

Finishes writing any remaining data to the OutputStream then closes the stream.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.315.3.2 `virtual void decaf::util::zip::DeflaterOutputStream::deflate ( ) throw ( decaf::io::IOException )` [protected, virtual]

Writes a buffers worth of compressed data to the wrapped OutputStream.

6.315.3.3 `virtual void decaf::util::zip::DeflaterOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.315.3.4 `virtual void decaf::util::zip::DeflaterOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` `[protected, virtual]`

Reimplemented from **decaf::io::FilterOutputStream** (p. 1863).

6.315.3.5 `virtual void decaf::util::zip::DeflaterOutputStream::finish ( ) throw ( decaf::io::IOException )` `[virtual]`

Finishes writing any remaining data to the wrapped OutputStream but does not close it upon completion.

#### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

#### 6.315.4 Field Documentation

6.315.4.1 `std::vector<unsigned char> decaf::util::zip::DeflaterOutputStream::buf`  
`[protected]`

The Buffer to use for.

6.315.4.2 `const std::size_t decaf::util::zip::DeflaterOutputStream::DEFAULT_BUFFER_SIZE` `[static, protected]`

6.315.4.3 `Deflater* decaf::util::zip::DeflaterOutputStream::deflater`  
`[protected]`

The **Deflater** (p. 1672) for this stream.

6.315.4.4 `bool decaf::util::zip::DeflaterOutputStream::isDone` `[protected]`

6.315.4.5 `bool decaf::util::zip::DeflaterOutputStream::ownDeflater`  
`[protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/DeflaterOutputStream.h`

## 6.316 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

```
#include <src/main/decaf/util/concurrent/Delayed.h>
```

Inheritance diagram for decaf::util::concurrent::Delayed:

### Public Member Functions

- virtual **~Delayed** ()
- virtual long long **getDelay** (const **TimeUnit** &unit)=0  
*Returns the remaining delay associated with this object, in the given time unit.*

#### 6.316.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay.

An implementation of this interface must define a Comparable methods that provides an ordering consistent with its getDelay method.

#### 6.316.2 Constructor & Destructor Documentation

6.316.2.1 virtual decaf::util::concurrent::Delayed::~Delayed ( ) [inline, virtual]

#### 6.316.3 Member Function Documentation

6.316.3.1 virtual long long decaf::util::concurrent::Delayed::getDelay ( const **TimeUnit** & *unit* )  
[pure virtual]

Returns the remaining delay associated with this object, in the given time unit.

#### Parameters

<i>unit</i>	The time unit
-------------	---------------

#### Returns

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Delayed.h**

## 6.317 cms::DeliveryMode Class Reference

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

```
#include <src/main/cms/DeliveryMode.h>
```

### Public Types

- enum **DELIVERY\_MODE** { **PERSISTENT** = 0, **NON\_PERSISTENT** = 1 }

*Enumeration values for **Message** (p. 2493) Delivery Mode.*

### Public Member Functions

- virtual **~DeliveryMode** ()

#### 6.317.1 Detailed Description

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

When a client sends a **cms : Message** (p. 2493) it can mark the **Message** (p. 2493) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 2493) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 2493) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 2493) throughput.

The **DeliveryMode** (p. 1687) covers only the transport of the **Message** (p. 2493) for sending client to its destination and doesn't apply to the receiving **Message** (p. 2493) consumer. The receiving Consumer can drop Message's based on configuration such as memory limits or **Message** (p. 2493) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is PERSISTENT and the configuration of the **Message** (p. 2493) consumer allows for it.

#### Since

1.0

#### 6.317.2 Member Enumeration Documentation

##### 6.317.2.1 enum cms::DeliveryMode::DELIVERY\_MODE

Enumeration values for **Message** (p. 2493) Delivery Mode.

Enumerator:

**PERSISTENT**

**NON\_PERSISTENT**

### 6.317.3 Constructor & Destructor Documentation

6.317.3.1 `virtual cms::DeliveryMode::~~DeliveryMode ( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/DeliveryMode.h`

## 6.318 cms::Destination Class Reference

A **Destination** (p. 1688) object encapsulates a provider-specific address.

```
#include <src/main/cms/Destination.h>
```

Inheritance diagram for cms::Destination:

### Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY\_TOPIC**, **TEMPORARY\_QUEUE** }

### Public Member Functions

- virtual **~Destination** ()
- virtual **DestinationType** **getDestinationType** () const =0  
*Retrieve the **Destination** (p. 1688) Type for this **Destination** (p. 1688).*
- virtual **cms::Destination** \* **clone** () const =0  
*Creates a new instance of this destination type that is a copy of this one, and returns it.*
- virtual void **copy** (const **cms::Destination** &source)=0  
*Copies the contents of the given **Destination** (p. 1688) object to this one.*
- virtual const **CMSProperties** & **getCMSProperties** () const =0  
*Retrieve any properties that might be part of the destination that was specified.*

### 6.318.1 Detailed Description

A **Destination** (p. 1688) object encapsulates a provider-specific address.

There is no standard definition of a **Destination** (p. 1688) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1688) address.

All CMS **Destination** (p. 1688) objects support concurrent use.

#### Since

1.0

### 6.318.2 Member Enumeration Documentation

6.318.2.1 enum cms::Destination::DestinationType

#### Enumerator:

**TOPIC**

**QUEUE**

**TEMPORARY\_TOPIC**

**TEMPORARY\_QUEUE**

### 6.318.3 Constructor & Destructor Documentation

6.318.3.1 virtual cms::Destination::~~Destination ( ) [inline, virtual]

### 6.318.4 Member Function Documentation

6.318.4.1 virtual cms::Destination\* cms::Destination::clone ( ) const [pure virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

#### Returns

cloned copy of this object

Implemented in **activemq::commands::ActiveMQQueue** (p. 454), **activemq::commands::ActiveMQTempQueue** (p. 575), **activemq::commands::ActiveMQTempTopic** (p. 603), and **activemq::commands::ActiveMQTopic** (p. 661).

6.318.4.2 virtual void cms::Destination::copy ( const cms::Destination & source ) [pure virtual]

Copies the contents of the given **Destination** (p. 1688) object to this one.

**Parameters**

<i>source</i>	The source <b>Destination</b> (p. 1688) object.
---------------	---

Implemented in **activemq::commands::ActiveMQQueue** (p. 455), **activemq::commands::ActiveMQTempQueue** (p. 576), **activemq::commands::ActiveMQTempTopic** (p. 604), and **activemq::commands::ActiveMQTopic** (p. 661).

**6.318.4.3** `virtual const CMSProperties& cms::Destination::getCMSProperties ( ) const`  
`[pure virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

**Returns**

A {const} reference to a **CMSProperties** (p. 1135) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 456), **activemq::commands::ActiveMQTempQueue** (p. 577), **activemq::commands::ActiveMQTempTopic** (p. 605), and **activemq::commands::ActiveMQTopic** (p. 662).

**6.318.4.4** `virtual DestinationType cms::Destination::getDestinationType ( ) const` `[pure virtual]`

Retrieve the **Destination** (p. 1688) Type for this **Destination** (p. 1688).

**Returns**

The **Destination** (p. 1688) Type

Implemented in **activemq::commands::ActiveMQQueue** (p. 456), **activemq::commands::ActiveMQTempQueue** (p. 577), **activemq::commands::ActiveMQTempTopic** (p. 606), and **activemq::commands::ActiveMQTopic** (p. 663).

The documentation for this class was generated from the following file:

- `src/main/cms/Destination.h`

## 6.319 **activemq::commands::ActiveMQDestination::DestinationFilter** **Struct Reference**

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

**Static Public Attributes**

- static const std::string **ANY\_CHILD**
- static const std::string **ANY\_DESCENDENT**



### 6.319.1 Field Documentation

6.319.1.1 `const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_CHILD` `[static]`

6.319.1.2 `const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_DESCENDENT` `[static]`

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

## 6.320 activemq::commands::DestinationInfo Class Reference

```
#include <src/main/activemq/commands/DestinationInfo.h>
```

Inheritance diagram for `activemq::commands::DestinationInfo`:

### Public Member Functions

- **DestinationInfo** ()
- virtual **~DestinationInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DestinationInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)

- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char **operationType**)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long **timeout**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_DESTINATIONINFO** = 8

### Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **ActiveMQDestination** > **destination**
- unsigned char **operationType**
- long long **timeout**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**

## 6.320.1 Constructor & Destructor Documentation

6.320.1.1 **activemq::commands::DestinationInfo::DestinationInfo** ( )

6.320.1.2 **virtual activemq::commands::DestinationInfo::~~DestinationInfo** ( ) [virtual]

## 6.320.2 Member Function Documentation

6.320.2.1 **virtual DestinationInfo\*** **activemq::commands::DestinationInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.320.2.2 `virtual void activemq::commands::DestinationInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.320.2.3 `virtual bool activemq::commands::DestinationInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.320.2.4 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ( ) const [virtual]`

6.320.2.5 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ( ) [virtual]`

6.320.2.6 `virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ( ) const [virtual]`

6.320.2.7 `virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ( ) [virtual]`

6.320.2.8 `virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

- 6.320.2.9 `virtual const Pointer<ActiveMQDestination>&  
activemq::commands::DestinationInfo::getDestination ( ) const` [virtual]
- 6.320.2.10 `virtual Pointer<ActiveMQDestination>&  
activemq::commands::DestinationInfo::getDestination ( )` [virtual]
- 6.320.2.11 `virtual unsigned char activemq::commands::DestinationInfo::getOperationType ( )  
const` [virtual]
- 6.320.2.12 `virtual long long activemq::commands::DestinationInfo::getTimeout ( ) const`  
[virtual]
- 6.320.2.13 `virtual void activemq::commands::DestinationInfo::setBrokerPath ( const  
std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath )`  
[virtual]
- 6.320.2.14 `virtual void activemq::commands::DestinationInfo::setConnectionId ( const  
Pointer< ConnectionId > & connectionId )` [virtual]
- 6.320.2.15 `virtual void activemq::commands::DestinationInfo::setDestination ( const Pointer<  
ActiveMQDestination > & destination )` [virtual]
- 6.320.2.16 `virtual void activemq::commands::DestinationInfo::setOperationType ( unsigned  
char operationType )` [virtual]
- 6.320.2.17 `virtual void activemq::commands::DestinationInfo::setTimeout ( long long timeout )`  
[virtual]
- 6.320.2.18 `virtual std::string activemq::commands::DestinationInfo::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

- 6.320.2.19 `virtual Pointer<Command> activemq::commands::DestinationInfo::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.320.3 Field Documentation

- 6.320.3.1 `std::vector< decaf::lang::Pointer<BrokerId> >`  
`activemq::commands::DestinationInfo::brokerPath` [protected]
- 6.320.3.2 `Pointer<ConnectionId>` `activemq::commands::DestinationInfo::connectionId`  
[protected]
- 6.320.3.3 `Pointer<ActiveMQDestination>` `activemq::commands::DestinationInfo::destination`  
[protected]
- 6.320.3.4 `const unsigned char` `activemq::commands::DestinationInfo::ID_DESTINATIONINFO = 8` [static]
- 6.320.3.5 `unsigned char` `activemq::commands::DestinationInfo::operationType`  
[protected]
- 6.320.3.6 `long long` `activemq::commands::DestinationInfo::timeout`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DestinationInfo.h`

## 6.321 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1696).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller`:

### Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual `~DestinationInfoMarshaller` ()

- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
**throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs)** **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn)** **throw ( decaf::io::IOException )**  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut)** **throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

### 6.321.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1696).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.321.2 Constructor & Destructor Documentation

- 6.321.2.1 **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::DestinationInfoMarshaller ( )** *[inline]*
- 6.321.2.2 **virtual activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::~~DestinationInfoMarshaller ( )** *[inline, virtual]*

### 6.321.3 Member Function Documentation

- 6.321.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::createObject ( )** **const** *[virtual]*

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

**6.321.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.321.3.3** `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

**6.321.3.4** `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.321.3.5  virtual int activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.321.3.6  virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.322 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller

### Class Reference

1707

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

6.321.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DestinationInfoMarshaller.h**

## 6.322 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1700).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller:

#### Public Member Functions

- **DestinationInfoMarshaller** ()

- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.322.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1700).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.322.2 Constructor & Destructor Documentation

6.322.2.1 **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::DestinationInfoMarshaller** ( ) [inline]

6.322.2.2 **virtual activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::~~DestinationInfoMarshaller** ( ) [inline, virtual]

### 6.322.3 Member Function Documentation

6.322.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.322.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.322.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.322.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.322.3.5  virtual int activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.322.3.6  virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.323 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller

### Class Reference

1711

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

6.322.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h**

## 6.323 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1704).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller:

#### Public Member Functions

- **DestinationInfoMarshaller** ()

- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.323.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1704).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.323.2 Constructor & Destructor Documentation

- 6.323.2.1 **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::DestinationInfoMarshaller** ( ) [inline]
- 6.323.2.2 **virtual activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::~~DestinationInfoMarshaller** ( ) [inline, virtual]

### 6.323.3 Member Function Documentation

- 6.323.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.323.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::getDataStructureType  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.323.3.3 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseMarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \*  
 dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.323.3.4 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseUnmarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,  
 decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.323.3.5  virtual int activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.323.3.6  virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

6.323.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DestinationInfoMarshaller.h**

6.324 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller  
Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1708).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()

- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.324.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1708).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.324.2 Constructor & Destructor Documentation

- 6.324.2.1 **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::DestinationInfoMarshaller** ( ) [inline]
- 6.324.2.2 **virtual activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::~~DestinationInfoMarshaller** ( ) [inline, virtual]

### 6.324.3 Member Function Documentation

- 6.324.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

**6.324.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.324.3.3** `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

**6.324.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.324.3.5  virtual int activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.324.3.6  virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.325 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller

### Class Reference

1719

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

6.324.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DestinationInfoMarshaller.h**

## 6.325 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1712).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller:

#### Public Member Functions

- **DestinationInfoMarshaller** ()

- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.325.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1712).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.325.2 Constructor & Destructor Documentation

- 6.325.2.1 **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::DestinationInfoMarshaller**  
 ( ) [inline]
- 6.325.2.2 **virtual activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::~~DestinationInfoMarshaller**  
 ( ) [inline, virtual]

### 6.325.3 Member Function Documentation

- 6.325.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.325.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.325.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.325.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.325.3.5  virtual int activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.325.3.6  virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.326 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller

### Class Reference

1723

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

6.325.3.7 virtual void activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DestinationInfoMarshaller.h**

## 6.326 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1716).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller:

#### Public Member Functions

- **DestinationInfoMarshaller** ()

- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.326.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1716).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.326.2 Constructor & Destructor Documentation

- 6.326.2.1 **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::DestinationInfoMarshaller** ( ) [inline]
- 6.326.2.2 **virtual activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::~~DestinationInfoMarshaller** ( ) [inline, virtual]

### 6.326.3 Member Function Documentation

- 6.326.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.326.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::getDataStructureType  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.326.3.3 virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseMarshal  
( OpenWireFormat \* wireFormat, commands::DataStructure \*  
dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.326.3.4 virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseUnmarshal  
( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,  
decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )  
[virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.326.3.5  virtual int activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.326.3.6  virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.326.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DestinationInfoMarshaller.h**

**6.327 activemq::cmsutil::DestinationResolver Class Reference**

Resolves a CMS destination name to a *Destination*.

```
#include <src/main/activemq/cmsutil/DestinationResolver.h>
```

Inheritance diagram for **activemq::cmsutil::DestinationResolver**:

**Public Member Functions**

- virtual **~DestinationResolver** ()
- virtual void **init** (**ResourceLifecycleManager** \*mgr)=0  
*Initializes this destination resolver for use.*

- virtual void **destroy** ()=0  
*Destroys any allocated resources.*
- virtual **cms::Destination** \* **resolveDestinationName** (**cms::Session** \*session, const std::string &destName, bool pubSubDomain)=0 throw ( cms::CMSException )  
*Resolves the given name to a destination.*

### 6.327.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

### 6.327.2 Constructor & Destructor Documentation

6.327.2.1 virtual **activemq::cmsutil::DestinationResolver::~~DestinationResolver** ( )  
[inline, virtual]

### 6.327.3 Member Function Documentation

6.327.3.1 virtual void **activemq::cmsutil::DestinationResolver::destroy** ( ) [pure virtual]

Destroys any allocated resources.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1787).

6.327.3.2 virtual void **activemq::cmsutil::DestinationResolver::init** (  
**ResourceLifecycleManager** \* mgr ) [pure virtual]

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p. 1721)).

#### Parameters

<i>mgr</i>	the resource lifecycle manager.
------------	---------------------------------

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1787).

6.327.3.3 virtual **cms::Destination**\* **activemq::cmsutil::DestinationResolver::resolveDestinationName** (  
**cms::Session** \* session, const std::string & destName, bool pubSubDomain )  
throw ( **cms::CMSException** ) [pure virtual]

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

**Parameters**

<i>session</i>	the session for which to retrieve resolve the destination.
<i>destName</i>	the name to be resolved.
<i>pubSubDomain</i>	If true, the name will be resolved to a Topic, otherwise a Queue.

**Returns**

the resolved destination

**Exceptions**

<b>cms::CMSException</b> (p. 1130)	if resolution failed.
---------------------------------------	-----------------------

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1788).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DestinationResolver.h**

**6.328 activemq::commands::DiscoveryEvent Class Reference**

```
#include <src/main/activemq/commands/DiscoveryEvent.h>
```

Inheritance diagram for activemq::commands::DiscoveryEvent:

**Public Member Functions**

- **DiscoveryEvent** ()
- virtual **~DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DiscoveryEvent \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &**serviceName**)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &**brokerName**)

### Static Public Attributes

- static const unsigned char **ID\_DISCOVERYEVENT** = 40

### Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

## 6.328.1 Constructor & Destructor Documentation

6.328.1.1 `activemq::commands::DiscoveryEvent::DiscoveryEvent ( )`

6.328.1.2 `virtual activemq::commands::DiscoveryEvent::~~DiscoveryEvent ( ) [virtual]`

## 6.328.2 Member Function Documentation

6.328.2.1 `virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1628).

6.328.2.2 `virtual void activemq::commands::DiscoveryEvent::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------



Implements **activemq::commands::DataStructure** (p. 1629).

6.328.2.3 `virtual bool activemq::commands::DiscoveryEvent::equals ( const DataStructure *  
value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.328.2.4 `virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName ( )  
const [virtual]`

6.328.2.5 `virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName ( )  
[virtual]`

6.328.2.6 `virtual unsigned char activemq::commands::DiscoveryEvent::getDataStructureType ( )  
const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.328.2.7 `virtual std::string& activemq::commands::DiscoveryEvent::getServiceName ( )  
[virtual]`

6.328.2.8 `virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName ( )  
const [virtual]`

6.328.2.9 `virtual void activemq::commands::DiscoveryEvent::setBrokerName ( const std::string  
& brokerName ) [virtual]`

6.328.2.10 `virtual void activemq::commands::DiscoveryEvent::setServiceName ( const  
std::string & serviceName ) [virtual]`

6.328.2.11 `virtual std::string activemq::commands::DiscoveryEvent::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 796).

### 6.328.3 Field Documentation

6.328.3.1 `std::string activemq::commands::DiscoveryEvent::brokerName`  
`[protected]`

6.328.3.2 `const unsigned char activemq::commands::DiscoveryEvent::ID_ -`  
`DISCOVERYEVENT = 40 [static]`

6.328.3.3 `std::string activemq::commands::DiscoveryEvent::serviceName`  
`[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DiscoveryEvent.h`

## 6.329 **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller** Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1725).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventM
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller**:

### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataSet \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataSetType** () const

*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.329.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1725).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.329.2 Constructor & Destructor Documentation

6.329.2.1 **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [inline]

6.329.2.2 **virtual activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::~DiscoveryEventMarshaller**  
( ) [inline, virtual]

### 6.329.3 Member Function Documentation

6.329.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.329.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.329.3.3  virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.329.3.4  virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.329.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.329.3.6 `virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.329.3.7 virtual void activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**DiscoveryEventMarshaller.h**

## 6.330 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1729).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventM
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.330.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1729).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.330.2 Constructor & Destructor Documentation

6.330.2.1 **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
 ( ) [**inline**]

6.330.2.2 **virtual activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
 ( ) [**inline**, **virtual**]

### 6.330.3 Member Function Documentation

6.330.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::createObject** (  
 ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.330.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.330.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.330.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



### 6.330 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller

#### Class Reference 1739

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.330.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.330.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.330.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DiscoveryEventMarshaller.h**

## 6.331 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1733).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventM
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.331.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1733).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.331.2 Constructor & Destructor Documentation

6.331.2.1 **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [inline]

6.331.2.2 **virtual activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
( ) [inline, virtual]

### 6.331.3 Member Function Documentation

6.331.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.331.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.331.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.331.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

### 6.331 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller

#### Class Reference 1743

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.331.3.5  virtual int activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.331.3.6  virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```

6.331.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DiscoveryEventMarshaller.h**

## 6.332 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1737).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventM
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.332.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1737).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.332.2 Constructor & Destructor Documentation

6.332.2.1 **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
 ( ) [inline]

6.332.2.2 **virtual activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
 ( ) [inline, virtual]

### 6.332.3 Member Function Documentation

6.332.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.332.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.332.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.332.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



### 6.332 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller

#### Class Reference 1747

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.332.3.5 virtual int activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.332.3.6 virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.332.3.7 virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**DiscoveryEventMarshaller.h**

## 6.333 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1741).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventM
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.333.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1741).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.333.2 Constructor & Destructor Documentation

6.333.2.1 **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
 ( ) [**inline**]

6.333.2.2 **virtual activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
 ( ) [**inline**, **virtual**]

### 6.333.3 Member Function Documentation

6.333.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::createObject** ( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.333.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.333.3.3  virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.333.3.4  virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

### 6.333 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller

#### Class Reference 1751

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.333.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.333.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.333.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DiscoveryEventMarshaller.h**

## 6.334 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1745).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventM
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller:

#### Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.334.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1745).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.334.2 Constructor & Destructor Documentation

6.334.2.1 **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::DiscoveryEventMarshaller**  
( ) [**inline**]

6.334.2.2 **virtual activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller**  
( ) [**inline**, **virtual**]

### 6.334.3 Member Function Documentation

6.334.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::createObject** ( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.334.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.334.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.334.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



### 6.334 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller

#### Class Reference 1755

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.334.3.5 virtual int activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.334.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.334.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**DiscoveryEventMarshaller.h**

## 6.335 activemq::core::DispatchData Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

```
#include <src/main/activemq/core/DispatchData.h>
```

### Public Member Functions

- **DispatchData** ()
- **DispatchData** (const **decaf::lang::Pointer**< **commands::ConsumerId** > &consumer, const **decaf::lang::Pointer**< **commands::Message** > &message)
- const **decaf::lang::Pointer**< **commands::ConsumerId** > & **getConsumerId** ()
- const **decaf::lang::Pointer**< **commands::Message** > & **getMessage** ()

### 6.335.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

### 6.335.2 Constructor & Destructor Documentation

6.335.2.1 `activemq::core::DispatchData::DispatchData ( )` `[inline]`

6.335.2.2 `activemq::core::DispatchData::DispatchData ( const decaf::lang::Pointer< commands::ConsumerId > & consumer, const decaf::lang::Pointer< commands::Message > & message )` `[inline]`

### 6.335.3 Member Function Documentation

6.335.3.1 `const decaf::lang::Pointer< commands::ConsumerId > & activemq::core::DispatchData::getConsumerId ( )` `[inline]`

6.335.3.2 `const decaf::lang::Pointer< commands::Message > & activemq::core::DispatchData::getMessage ( )` `[inline]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/DispatchData.h`

## 6.336 activemq::core::Dispatcher Class Reference

Interface for an object responsible for dispatching messages to consumers.

```
#include <src/main/activemq/core/Dispatcher.h>
```

Inheritance diagram for `activemq::core::Dispatcher`:

### Public Member Functions

- `virtual ~Dispatcher ( )`
- `virtual void dispatch (const Pointer< MessageDispatch > &message)=0`  
*Dispatches a message to a particular consumer.*

### 6.336.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

### 6.336.2 Constructor & Destructor Documentation

6.336.2.1 `virtual activemq::core::Dispatcher::~Dispatcher ( )` `[inline, virtual]`

### 6.336.3 Member Function Documentation

6.336.3.1 `virtual void activemq::core::Dispatcher::dispatch ( const Pointer< MessageDispatch > & message ) [pure virtual]`

Dispatches a message to a particular consumer.

#### Parameters

<i>message</i>	- the message to be dispatched.
----------------	---------------------------------

Implemented in `activemq::core::ActiveMQConsumer` (p. 287), and `activemq::core::ActiveMQSession` (p. 496).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Dispatcher.h`

## 6.337 decaf::lang::Double Class Reference

```
#include <src/main/decaf/lang/Double.h>
```

Inheritance diagram for `decaf::lang::Double`:

#### Public Member Functions

- **Double** (double value)
- **Double** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual **~Double** ()
- virtual int **compareTo** (const **Double** &d) const  
*Compares this **Double** (p. 1751) instance with another.*
- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Double** &d) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const double &d) const  
*Compares this **Double** (p. 1751) instance with another.*
- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const double &d) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const

- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*
- bool **isInfinite** () const
- bool **isNaN** () const

### Static Public Member Functions

- static int **compare** (double d1, double d2)  
*Compares the two specified double values.*
- static long long **doubleToLongBits** (double value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.*
- static long long **doubleToRawLongBits** (double value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.*
- static bool **isInfinite** (double value)
- static bool **isNaN** (double value)
- static double **longBitsToDouble** (long long bits)  
*Returns the double value corresponding to a given bit representation.*
- static double **parseDouble** (const std::string value) throw ( exceptions::NumberFormatException )  
*Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 1751).*
- static std::string **toHexString** (double value)  
*Returns a hexadecimal string representation of the double argument.*
- static std::string **toString** (double value)  
*Returns a string representation of the double argument.*
- static **Double** **valueOf** (double value)  
*Returns a **Double** (p. 1751) instance representing the specified double value.*
- static **Double** **valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Double** (p. 1751) instance that wraps a primitive double which is parsed from the string value passed.*

## Static Public Attributes

- static const int **SIZE** = 64  
*The size in bits of the primitive int type.*
- static const double **MAX\_VALUE**  
*The maximum value that the primitive type can hold.*
- static const double **MIN\_VALUE**  
*The minimum value that the primitive type can hold.*
- static const double **NaN**  
*Constant for the Not a **Number** (p. 2786) Value.*
- static const double **POSITIVE\_INFINITY**  
*Constant for Positive Infinity.*
- static const double **NEGATIVE\_INFINITY**  
*Constant for Negative Infinity.*

## 6.337.1 Constructor & Destructor Documentation

### 6.337.1.1 `decaf::lang::Double::Double ( double value )`

#### Parameters

<code>value</code>	- the primitive type to wrap
--------------------	------------------------------

### 6.337.1.2 `decaf::lang::Double::Double ( const std::string & value ) throw ( exceptions::NumberFormatException )`

#### Parameters

<code>value</code>	- the string to convert to a primitive type to wrap
--------------------	---

### 6.337.1.3 `virtual decaf::lang::Double::~~Double ( ) [inline, virtual]`

## 6.337.2 Member Function Documentation

### 6.337.2.1 `virtual unsigned char decaf::lang::Double::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

#### Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2787).

6.337.2.2 static int decaf::lang::Double::compare ( double *d1*, double *d2* ) [static]

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: new Double(*d1*).compareTo(new Double(*d2*))

#### Parameters

<i>d1</i>	- the first double to compare
<i>d2</i>	- the second double to compare

#### Returns

the value 0 if *d1* is numerically equal to *d2*; a value less than 0 if *d1* is numerically less than *d2*; and a value greater than 0 if *d1* is numerically greater than *d2*.

6.337.2.3 virtual int decaf::lang::Double::compareTo ( const double & *d* ) const  
[virtual]

Compares this **Double** (p. 1751) instance with another.

#### Parameters

<i>d</i>	- the <b>Double</b> (p. 1751) instance to be compared
----------	---

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< double > (p. 1187).

6.337.2.4 virtual int decaf::lang::Double::compareTo ( const Double & *d* ) const  
[virtual]

Compares this **Double** (p. 1751) instance with another.

#### Parameters

<i>d</i>	- the <b>Double</b> (p. 1751) instance to be compared
----------	---

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< Double > (p. 1187).

**6.337.2.5** `static long long decaf::lang::Double::doubleToLongBits ( double value )`  
`[static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.

Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000ffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToLongBits` (except all NaN values are collapsed to a single "canonical" NaN value).

#### Parameters

<i>value</i>	- double to be converted
--------------	--------------------------

#### Returns

the long long bits that make up the double

**6.337.2.6** `static long long decaf::lang::Double::doubleToRawLongBits ( double value )`  
`[static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.

Bit 63 (the bit that is selected by the mask 0x8000000000000000LL) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000ffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000LL. If the argument is negative infinity, the result is 0xfff0000000000000LL. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the `doubleToLongBits` method, `doubleToRawLongBits` does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the `longBitsToDouble(long)` method, will produce a floating-point value the same as the argument to `doubleToRawLongBits`.

#### Parameters

<i>value</i>	- double to be converted
--------------	--------------------------



**Returns**

the long long bits that make up the double

```
6.337.2.7 virtual double decaf::lang::Double::doubleValue ( ) const [inline, virtual]
```

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

```
6.337.2.8 bool decaf::lang::Double::equals ( const Double & d ) const [inline, virtual]
```

**Parameters**

<i>d</i> - the <b>Double</b> (p. 1751) object to compare against.
---

**Returns**

true if the two **Double** (p. 1751) Objects have the same value.

Implements **decaf::lang::Comparable**< **Double** > (p. 1188).

```
6.337.2.9 bool decaf::lang::Double::equals ( const double & d ) const [inline, virtual]
```

**Parameters**

<i>d</i> - the <b>Double</b> (p. 1751) object to compare against.
---

**Returns**

true if the two **Double** (p. 1751) Objects have the same value.

Implements **decaf::lang::Comparable**< **double** > (p. 1188).

```
6.337.2.10 virtual float decaf::lang::Double::floatValue ( ) const [inline, virtual]
```

Answers the float value which the receiver represents.

**Returns**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.337.2.11 `virtual int decaf::lang::Double::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.337.2.12 `bool decaf::lang::Double::isInfinite ( ) const`

#### Returns

true if the double is equal to positive infinity.

6.337.2.13 `static bool decaf::lang::Double::isInfinite ( double value )` `[static]`

#### Parameters

<i>value</i>	- The double to check.
--------------	------------------------

#### Returns

true if the double is equal to infinity.

6.337.2.14 `bool decaf::lang::Double::isNaN ( ) const`

#### Returns

true if the double is equal to NaN.

6.337.2.15 `static bool decaf::lang::Double::isNaN ( double value )` `[static]`

#### Parameters

<i>value</i>	- The double to check.
--------------	------------------------

#### Returns

true if the double is equal to NaN.

6.337.2.16 `static double decaf::lang::Double::longBitsToDouble ( long long bits )`  
`[static]`

Returns the double value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xfffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1755) method.

#### Parameters

<i>bits</i>	- the long long bits to convert to double
-------------	---

#### Returns

the double converted from the bits

```
6.337.2.17 virtual long long decaf::lang::Double::longValue ( ) const [inline,
virtual]
```

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

```
6.337.2.18 virtual bool decaf::lang::Double::operator< ( const Double & d ) const
[inline, virtual]
```

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Double >** (p. 1188).

**6.337.2.19** `virtual bool decaf::lang::Double::operator< ( const double & d ) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **double** > (p. 1188).

**6.337.2.20** `virtual bool decaf::lang::Double::operator== ( const Double & d ) const`  
[inline, virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Double** > (p. 1189).

**6.337.2.21** `virtual bool decaf::lang::Double::operator== ( const double & d ) const`  
[inline, virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>d</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **double** > (p. 1189).

6.337.2.22 static double decaf::lang::Double::parseDouble ( const std::string *value* ) throw ( exceptions::NumberFormatException ) [static]

Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 1751).

#### Parameters

<i>value</i>	- The string to parse to an double
--------------	------------------------------------

#### Returns

a double parsed from the passed string

#### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

6.337.2.23 virtual short decaf::lang::Double::shortValue ( ) const [inline, virtual]

Answers the short value which the receiver represents.

#### Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2788).

6.337.2.24 static std::string decaf::lang::Double::toHexString ( double *value* ) [static]

Returns a hexadecimal string representation of the double argument.

All characters mentioned below are ASCII characters.

\* If the argument is NaN, the result is the string "NaN". \* Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 2051) on the exponent value. o If m is

a double value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

#### Parameters

<i>value</i>	- The double to convert to a string
--------------	-------------------------------------

#### Returns

the Hex formatted double string.

6.337.2.25 `std::string decaf::lang::Double::toString ( ) const`

#### Returns

this **Double** (p. 1751) Object as a **String** (p. 3610) Representation

6.337.2.26 `static std::string decaf::lang::Double::toString ( double value ) [static]`

Returns a string representation of the double argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude *m*:  
 o If *m* is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".  
 o If *m* is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".  
 o If *m* is greater than or equal to 10<sup>-3</sup> but less than 10<sup>7</sup>, then it is represented as the integer part of *m*, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of *m*.  
 o If *m* is less than 10<sup>-3</sup> or greater than or equal to 10<sup>7</sup>, then it is represented in so-called "computerized scientific notation." Let *n* be the unique integer such that 10<sup>*n*</sup> ≤ *m* < 10<sup>*n*+1</sup>; then let *a* be the mathematically exact quotient of *m* and 10<sup>*n*</sup> so that 1 ≤ *a* < 10. The magnitude is then represented as the integer part of *a*, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of *a*, followed by the letter 'E', followed by a representation of *n* as a decimal integer, as produced by the method **Integer.toString(int)** (p. 2052).

#### Parameters

<i>value</i>	- The double to convert to a string
--------------	-------------------------------------

#### Returns

the formatted double string.

6.337.2.27 static **Double** decaf::lang::Double::valueOf ( double *value* ) [static]

Returns a **Double** (p. 1751) instance representing the specified double value.

#### Parameters

<i>value</i>	- double to wrap
--------------	------------------

#### Returns

new **Double** (p. 1751) instance wrapping the primitive value

6.337.2.28 static **Double** decaf::lang::Double::valueOf ( const std::string & *value* ) throw ( exceptions::NumberFormatException ) [static]

Returns a **Double** (p. 1751) instance that wraps a primitive double which is parsed from the string value passed.

#### Parameters

<i>value</i>	- the string to parse
--------------	-----------------------

#### Returns

a new **Double** (p. 1751) instance wrapping the double parsed from value

#### Exceptions

<i>NumberFormatException</i>	on error.
------------------------------	-----------

### 6.337.3 Field Documentation

6.337.3.1 const double decaf::lang::Double::MAX\_VALUE [static]

The maximum value that the primitive type can hold.

6.337.3.2 const double decaf::lang::Double::MIN\_VALUE [static]

The minimum value that the primitive type can hold.

6.337.3.3 const double decaf::lang::Double::NaN [static]

Constant for the Not a **Number** (p. 2786) Value.

6.337.3.4 `const double decaf::lang::Double::NEGATIVE_INFINITY` `[static]`

Constant for Negative Infinity.

6.337.3.5 `const double decaf::lang::Double::POSITIVE_INFINITY` `[static]`

Constant for Positive Infinity.

6.337.3.6 `const int decaf::lang::Double::SIZE = 64` `[static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Double.h`

## 6.338 decaf::internal::nio::DoubleArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::DoubleArrayBuffer:

### Public Member Functions

- **DoubleArrayBuffer** (int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **DoubleArrayBuffer** (p. 1762) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **DoubleArrayBuffer** (double \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a **DoubleArrayBuffer** (p. 1762) object that wraps the given array.*
- **DoubleArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*
- **DoubleArrayBuffer** (const **DoubleArrayBuffer** &other)  
*Create a **DoubleArrayBuffer** (p. 1762) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ~**DoubleArrayBuffer** ()
- virtual double \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )



Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 887).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual DoubleBuffer \* **asReadOnlyBuffer** () const

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only double buffer which the caller then owns.

- virtual DoubleBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **DoubleBuffer** (p. 1773).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

- virtual **DoubleBuffer \* duplicate ()**

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new double **Buffer** (p. 887) which the caller owns.

- virtual double **get ()** throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the double at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

- virtual double **get (int index)** const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the value at the given index.

#### Parameters

index	The index in the <b>Buffer</b> (p. 887) where the double is to be read.
-------	---

#### Returns

the double that is located at the given index.

#### Exceptions

<b>IndexOutOfBoundsException</b>	if index is not smaller than the buffer's limit
----------------------------------	---

- virtual bool **hasArray ()** const

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual DoubleBuffer & **put** (double value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given doubles into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The doubles value to be written.</i>
-------	---

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	<i>if this buffer's current position is not smaller than its limit.</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only.</i>

- virtual DoubleBuffer & **put** (int index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given doubles into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 887) to write the data.</i>
value	<i>The doubles to write.</i>

**Returns**

*a reference to this buffer*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written, or the index is negative.</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only.</i>

- virtual DoubleBuffer \* **slice** () const

*Creates a new **DoubleBuffer** (p. 1773) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

the newly create **DoubleBuffer** (p. 1773) which the caller owns.

**Protected Member Functions**

- virtual void **setReadOnly** (bool value)

Sets this **DoubleArrayBuffer** (p. 1762) as Read-Only or not Read-Only.

**6.338.1 Constructor & Destructor Documentation**

6.338.1.1 **decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer**  
 ( int *capacity*, bool *readOnly* = false ) throw ( **decaf::lang::exceptions::IllegalArgumentException** )

Creates a **DoubleArrayBuffer** (p. 1762) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

**Parameters**

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

**Exceptions**

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

6.338.1.2 **decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer** ( double \* *array*, int *size*, int *offset*, int *length*, bool *readOnly* = false ) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException** )

Creates a **DoubleArrayBuffer** (p. 1762) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

**Parameters**

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL
-----------------------------	-------------------

<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.
----------------------------------	---

6.338.1.3 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed `ByteArrayAdapter` and start at the given offset.

The capacity and limit of the new **DoubleArrayBuffer** (p. 1762) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The <code>ByteArrayAdapter</code> to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.338.1.4 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer ( const DoubleArrayBuffer & other )`

Create a **DoubleArrayBuffer** (p. 1762) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayAdapter` and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>DoubleArrayBuffer</b> (p. 1762) this one is to mirror.
--------------	---

6.338.1.5 `virtual decaf::internal::nio::DoubleArrayBuffer::~~DoubleArrayBuffer ( )`  
[virtual]

## 6.338.2 Member Function Documentation

6.338.2.1 `virtual double* decaf::internal::nio::DoubleArrayBuffer::array ( ) throw  
( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 887).

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1776).

6.338.2.2 `virtual int decaf::internal::nio::DoubleArrayBuffer::arrayOffset ( ) throw  
( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1777).

6.338.2.3 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::asReadOnlyBuffer ( )`  
`const [virtual]`

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1777).

6.338.2.4 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact ( ) throw (`  
`decaf::nio::ReadOnlyBufferException ) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **DoubleBuffer** (p. 1773).

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::DoubleBuffer** (p. 1778).

6.338.2.5 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::duplicate ( )`  
`[virtual]`

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new double **Buffer** (p. 887) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1778).

6.338.2.6 `virtual double decaf::internal::nio::DoubleArrayBuffer::get ( ) throw (`  
`decaf::nio::BufferUnderflowException ) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the double at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return.
--	----------------------------------

Implements **decaf::nio::DoubleBuffer** (p. 1780).

6.338.2.7 `virtual double decaf::internal::nio::DoubleArrayBuffer::get ( int index ) const throw (`  
`decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Absolute get method.

Reads the value at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the double is to be read.
--------------	---

#### Returns

the double that is located at the given index.



### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implements **decaf::nio::DoubleBuffer** (p. 1779).

**6.338.2.8** `virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray ( ) const`  
[inline, virtual]

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1781).

**6.338.2.9** `virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly ( ) const`  
[inline, virtual]

Tells whether or not this buffer is read-only.

### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 890).

**6.338.2.10** `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put ( int index, double value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given doubles into this buffer at the given index.

### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The doubles to write.

### Returns

a reference to this buffer

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1782).

6.338.2.11 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put ( double value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The doubles value to be written.
--------------	----------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::DoubleBuffer** (p. 1781).

6.338.2.12 `virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly ( bool value )` [inline, protected, virtual]

Sets this **DoubleArrayBuffer** (p. 1762) as Read-Only or not Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.338.2.13 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **DoubleBuffer** (p. 1773) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **DoubleBuffer** (p. 1773) which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1784).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/DoubleArrayBuffer.h`

## 6.339 decaf::nio::DoubleBuffer Class Reference

This class defines four categories of operations upon double buffers:

```
#include <src/main/decaf/nio/DoubleBuffer.h>
```

Inheritance diagram for `decaf::nio::DoubleBuffer`:

### Public Member Functions

- `virtual ~DoubleBuffer ()`
- `virtual std::string toString () const`
- `virtual double * array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the double array that backs this buffer (optional operation).*
- `virtual int arrayOffset ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- `virtual DoubleBuffer * asReadOnlyBuffer () const =0`  
*Creates a new, read-only double buffer that shares this buffer's content.*
- `virtual DoubleBuffer & compact ()=0 throw ( ReadOnlyBufferException )`  
*Compacts this buffer.*

- virtual **DoubleBuffer** \* **duplicate** ()=0  
*Creates a new double buffer that shares this buffer's content.*
- virtual double **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual double **get** (int index) const =0 throw ( lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **DoubleBuffer** & **get** (std::vector< double > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **DoubleBuffer** & **get** (double \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible double array.*
- **DoubleBuffer** & **put** (**DoubleBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the doubles remaining in the given source buffer into this buffer.*
- **DoubleBuffer** & **put** (const double \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers doubles into this buffer from the given source array.*
- **DoubleBuffer** & **put** (std::vector< double > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source doubles array into this buffer.*
- virtual **DoubleBuffer** & **put** (double value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given doubles into this buffer at the current position, and then increments the position.*
- virtual **DoubleBuffer** & **put** (int index, double value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given doubles into this buffer at the given index.*
- virtual **DoubleBuffer** \* **slice** () const =0  
*Creates a new **DoubleBuffer** (p. 1773) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **DoubleBuffer** &value) const
- virtual bool **equals** (const **DoubleBuffer** &value) const
- virtual bool **operator==** (const **DoubleBuffer** &value) const
- virtual bool **operator<** (const **DoubleBuffer** &value) const

## Static Public Member Functions

- static **DoubleBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new **DoubleBuffer** (p. 1773).*
- static **DoubleBuffer** \* **wrap** (double \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **DoubleBuffer** (p. 1773).*
- static **DoubleBuffer** \* **wrap** (std::vector< double > &buffer)  
*Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1773).*

## Protected Member Functions

- **DoubleBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **DoubleBuffer** (p. 1773) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.339.1 Detailed Description

This class defines four categories of operations upon double buffers:

o Absolute and relative get and put methods that read and write single doubles; o Relative bulk get methods that transfer contiguous sequences of doubles from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

Since

1.0

### 6.339.2 Constructor & Destructor Documentation

6.339.2.1 decaf::nio::DoubleBuffer::DoubleBuffer ( int *capacity* ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **DoubleBuffer** (p. 1773) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

**Parameters**

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 887) in doubles
-----------------	---

**Exceptions**

<i>IllegalArgumentEx-ception</i>	if capacity is negative.
----------------------------------	--------------------------

6.339.2.2 `virtual decaf::nio::DoubleBuffer::~~DoubleBuffer ( ) [inline, virtual]`

**6.339.3 Member Function Documentation**

6.339.3.1 `static DoubleBuffer* decaf::nio::DoubleBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Allocates a new **DoubleBuffer** (p. 1773).

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

**Parameters**

<i>capacity</i>	The size of the Double buffer in doubles.
-----------------	---

**Returns**

the **DoubleBuffer** (p. 1773) that was allocated, caller owns.

**Exceptions**

<i>IllegalArgumentException</i>	is the capacity value is negative.
---------------------------------	------------------------------------

6.339.3.2 `virtual double* decaf::nio::DoubleBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 887).

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1768).

**6.339.3.3** `virtual int decaf::nio::DoubleBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1768).

**6.339.3.4** `virtual DoubleBuffer* decaf::nio::DoubleBuffer::asReadOnlyBuffer ( ) const [pure virtual]`

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns**

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1769).

**6.339.3.5** `virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact ( ) throw (   
 ReadOnlyBufferException ) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **DoubleBuffer** (p. 1773).

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1769).

**6.339.3.6** `virtual int decaf::nio::DoubleBuffer::compareTo ( const DoubleBuffer & value )   
 const [virtual]`

**6.339.3.7** `virtual DoubleBuffer* decaf::nio::DoubleBuffer::duplicate ( ) [pure   
 virtual]`

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new double **Buffer** (p. 887) which the caller owns.



Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1770).

6.339.3.8 `virtual bool decaf::nio::DoubleBuffer::equals ( const DoubleBuffer & value ) const`  
[virtual]

6.339.3.9 `DoubleBuffer& decaf::nio::DoubleBuffer::get ( std::vector< double > buffer )`  
`throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this **Buffer** (p. 887).

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length doubles remaining in this buffer
--	---

6.339.3.10 `virtual double decaf::nio::DoubleBuffer::get ( int index ) const throw (`  
`lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the double is to be read.
--------------	---

### Returns

the double that is located at the given index.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1771).

6.339.3.11 **DoubleBuffer& decaf::nio::DoubleBuffer::get ( double \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **BufferUnderflowException**, **decaf::lang::exceptions::IndexOutOfBoundsException**, **decaf::lang::exceptions::NullPointerException** )**

Relative bulk get method.

This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 892), then no bytes are transferred and a **BufferUnderflowException** (p. 916) is thrown.

Otherwise, this method copies `length` doubles from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 887).

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there are fewer than <code>length</code> doubles remaining in this buffer
<b>NullPointerException</b>	if the passed buffer is null.
<b>IndexOutOfBoundsException</b>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

6.339.3.12 **virtual double decaf::nio::DoubleBuffer::get ( ) throw ( **BufferUnderflowException** )** [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the double at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1770).

6.339.3.13 `virtual bool decaf::nio::DoubleBuffer::hasArray ( ) const [pure virtual]`

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

#### Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1771).

6.339.3.14 `virtual bool decaf::nio::DoubleBuffer::operator< ( const DoubleBuffer & value ) const [virtual]`

6.339.3.15 `virtual bool decaf::nio::DoubleBuffer::operator== ( const DoubleBuffer & value ) const [virtual]`

6.339.3.16 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put ( double value ) throw ( BufferOverflowException, ReadOnlyBufferException ) [pure virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

#### Parameters

<i>value</i>	The doubles value to be written.
--------------	----------------------------------

#### Returns

a reference to this buffer.

#### Exceptions

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1772).

6.339.3.17 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put ( int index, double value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException ) [pure virtual]`

Writes the given doubles into this buffer at the given index.

#### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The doubles to write.

#### Returns

a reference to this buffer

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<i>ReadOnlyBufferException</i> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1772).

6.339.3.18 `DoubleBuffer& decaf::nio::DoubleBuffer::put ( const double * buffer, int size, int offset, int length ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

This method transfers doubles into this buffer from the given source array.

If there are more doubles to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 892), then no doubles are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

#### Parameters

<i>buffer</i>	The array from which doubles are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of doubles to be read from the given array.

#### Returns

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if there is insufficient space in this buffer
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.339.3.19 **DoubleBuffer& decaf::nio::DoubleBuffer::put ( std::vector< double > & buffer )**  
 throw ( **BufferOverflowException**, **ReadOnlyBufferException** )

This method transfers the entire content of the given source doubles array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

**Parameters**

<i>buffer</i>	The buffer whose contents are copied to this <b>DoubleBuffer</b> (p. 1773).
---------------	---

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if there is insufficient space in this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

6.339.3.20 **DoubleBuffer& decaf::nio::DoubleBuffer::put ( DoubleBuffer & src )**  
 throw ( **BufferOverflowException**, **ReadOnlyBufferException**,  
 decaf::lang::exceptions::IllegalArgumentException )

This method transfers the doubles remaining in the given source buffer into this buffer.

If there are more doubles remaining in the source buffer than in this buffer, that is, if src.remaining() > **remaining()** (p. 892), then no doubles are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies n = src.remaining() doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

**Parameters**

<i>src</i>	The buffer to take doubles from an place in this one.
------------	---

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if there is insufficient space in this buffer for the remaining doubles in the source buffer
<b><i>IllegalArgumentException</i></b>	if the source buffer is this buffer.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

6.339.3.21 `virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice ( ) const` [pure virtual]

Creates a new **DoubleBuffer** (p. 1773) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

the newly create **DoubleBuffer** (p. 1773) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1773).

6.339.3.22 `virtual std::string decaf::nio::DoubleBuffer::toString ( ) const` [virtual]

**Returns**

a std::string describing this object

6.339.3.23 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap ( double * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **DoubleBuffer** (p. 1773).

The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **DoubleBuffer** (p. 1773) that is backed by `buffer`, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of <code>size</code> , <code>offset</code> , or <code>length</code> are not met.

**6.339.3.24** `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap ( std::vector< double > & buffer ) [static]`

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1773).

The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling <code>vector.resize( N )</code> .
---------------	--

#### Returns

a new **DoubleBuffer** (p. 1773) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/DoubleBuffer.h`

## 6.340 decaf::lang::DYNAMIC\_CAST\_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

### 6.341 `activemq::cmsutil::DynamicDestinationResolver` Class Reference

Resolves a CMS destination name to a `Destination`.

```
#include <src/main/activemq/cmsutil/DynamicDestinationResolver.h>
```

Inheritance diagram for `activemq::cmsutil::DynamicDestinationResolver`:

#### Data Structures

- class `SessionResolver`

*Manages maps of names to topics and queues for a single session.*

#### Public Member Functions

- `DynamicDestinationResolver ()`
- virtual `~DynamicDestinationResolver ()`
- virtual void `init (ResourceLifecycleManager *mgr)`  
*Initializes this destination resolver for use.*
- virtual void `destroy ()`  
*Destroys any allocated resources.*
- virtual `cms::Destination * resolveDestinationName (cms::Session *session, const std::string &destName, bool pubSubDomain) throw ( cms::CMSException )`  
*Resolves the given name to a destination.*

#### Protected Member Functions

- `DynamicDestinationResolver (const DynamicDestinationResolver &)`
- `DynamicDestinationResolver & operator= (const DynamicDestinationResolver &)`

#### 6.341.1 Detailed Description

Resolves a CMS destination name to a `Destination`.



### 6.341.2 Constructor & Destructor Documentation

6.341.2.1 `activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver ( const DynamicDestinationResolver & ) [inline, protected]`

6.341.2.2 `activemq::cmsutil::DynamicDestinationResolver::DynamicDestinationResolver ( )`

6.341.2.3 `virtual activemq::cmsutil::DynamicDestinationResolver::~~DynamicDestinationResolver ( ) [virtual]`

### 6.341.3 Member Function Documentation

6.341.3.1 `virtual void activemq::cmsutil::DynamicDestinationResolver::destroy ( ) [virtual]`

Destroys any allocated resources.

Implements **activemq::cmsutil::DestinationResolver** (p. 1721).

6.341.3.2 `virtual void activemq::cmsutil::DynamicDestinationResolver::init ( ResourceLifecycleManager * mgr ) [inline, virtual]`

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p. 1787)).

#### Parameters

<i>mgr</i>	the resource lifecycle manager.
------------	---------------------------------

Implements **activemq::cmsutil::DestinationResolver** (p. 1721).

6.341.3.3 `DynamicDestinationResolver& activemq::cmsutil::DynamicDestinationResolver::operator= ( const DynamicDestinationResolver & ) [inline, protected]`

6.341.3.4 `virtual cms::Destination* activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName ( cms::Session * session, const std::string & destName, bool pubSubDomain ) throw ( cms::CMSEException ) [virtual]`

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

#### Parameters

<i>session</i>	the session for which to retrieve resolve the destination.
<i>destName</i>	the name to be resolved.
<i>pubSubDomain</i>	If true, the name will be resolved to a Topic, otherwise a Queue.

**Returns**

the resolved destination

**Exceptions**

<b><i>cms::CMSEException</i></b> (p. 1130)	if resolution failed.
---	-----------------------

Implements **activemq::cmsutil::DestinationResolver** (p. 1721).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DynamicDestinationResolver.h**

## 6.342 decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference

```
#include <src/main/decaf/util/Map.h>
```

**Public Member Functions**

- **Entry** ()
- virtual **~Entry** ()
- virtual const K & **getKey** () const =0
- virtual const V & **getValue** () const =0
- virtual void **setValue** (const V &value)=0

```
template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::Map<
K, V, COMPARATOR >::Entry
```

### 6.342.1 Constructor & Destructor Documentation

6.342.1.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry ( ) [inline]`

6.342.1.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~~Entry ( ) [inline, virtual]`

### 6.342.2 Member Function Documentation

6.342.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey ( ) const [pure virtual]`

6.342.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue ( )  
const [pure virtual]`

6.342.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::Map< K, V, COMPARATOR >::Entry::setValue ( const V &  
value ) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

## 6.343 decaf::io::EOFException Class Reference

```
#include <src/main/decaf/io/EOFException.h>
```

Inheritance diagram for decaf::io::EOFException:

### Public Member Functions

- **EOFException** () throw ()  
*Default Constructor.*
- **EOFException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **EOFException** (const EOFException &ex) throw ()  
*Copy Constructor.*
- **EOFException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **EOFException** (const std::exception \*cause) throw ()  
*Constructor.*
- **EOFException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **EOFException** \* clone () const  
*Clones this exception.*
- virtual ~**EOFException** () throw ()

### 6.343.1 Constructor & Destructor Documentation

6.343.1.1 `decaf::io::EOFException::EOFException ( ) throw () [inline]`

Default Constructor.

6.343.1.2 `decaf::io::EOFException::EOFException ( const lang::Exception & ex ) throw ()`  
`[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.343.1.3 `decaf::io::EOFException::EOFException ( const EOFException & ex ) throw ()`  
`[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.343.1.4 `decaf::io::EOFException::EOFException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.343.1.5 `decaf::io::EOFException::EOFException ( const std::exception * cause ) throw ()`  
`[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.343.1.6 `decaf::io::EOFException::EOFException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.343.1.7 `virtual decaf::io::EOFException::~~EOFException ( ) throw () [inline, virtual]`

#### 6.343.2 Member Function Documentation

6.343.2.1 `virtual EOFException* decaf::io::EOFException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/EOFException.h`

## 6.344 decaf::util::logging::ErrorManager Class Reference

**ErrorManager** (p. 1792) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1941) during Logging.

```
#include <src/main/decaf/util/logging/ErrorManager.h>
```

#### Public Member Functions

- **ErrorManager** ()
- virtual **~ErrorMessage** ()
- virtual void **error** (const std::string &message, **decaf::lang::Exception** \*ex, int code)

*The error method is called when a **Handler** (p. 1941) failure occurs.*

## Static Public Attributes

- static const int **GENERIC\_FAILURE**  
*GENERIC\_FAILURE is used for failure that don't fit into one of the other categories.*
- static const int **WRITE\_FAILURE**  
*WRITE\_FAILURE is used when a write to an output stream fails.*
- static const int **FLUSH\_FAILURE**  
*FLUSH\_FAILURE is used when a flush to an output stream fails.*
- static const int **CLOSE\_FAILURE**  
*CLOSE\_FAILURE is used when a close of an output stream fails.*
- static const int **OPEN\_FAILURE**  
*OPEN\_FAILURE is used when an open of an output stream fails.*
- static const int **FORMAT\_FAILURE**  
*FORMAT\_FAILURE is used when formatting fails for any reason.*

### 6.344.1 Detailed Description

**ErrorManager** (p. 1792) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1941) during Logging.

When processing logging output, if a **Handler** (p. 1941) encounters problems then rather than throwing an Exception back to the issuer of the logging call (who is unlikely to be interested) the **Handler** (p. 1941) should call its associated **ErrorManager** (p. 1792).

#### Since

1.0

### 6.344.2 Constructor & Destructor Documentation

6.344.2.1 `decaf::util::logging::ErrorManager::ErrorManager ( )`

6.344.2.2 `virtual decaf::util::logging::ErrorManager::~ErrorManager ( )` [virtual]

### 6.344.3 Member Function Documentation

6.344.3.1 `virtual void decaf::util::logging::ErrorManager::error ( const std::string & message, decaf::lang::Exception * ex, int code )` [virtual]

The error method is called when a **Handler** (p. 1941) failure occurs.

This method may be overridden in subclasses. The default behavior in this base class is that the first call is reported to System.err, and subsequent calls are ignored.

#### Parameters

<i>msg</i>	- a descriptive string (may be empty)
<i>ex</i>	- an exception (may be NULL)
<i>code</i>	- an error code defined in <b>ErrorManager</b> (p. 1792)

#### 6.344.4 Field Documentation

6.344.4.1 `const int decaf::util::logging::ErrorManager::CLOSE_FAILURE`  
[static]

CLOSE\_FAILURE is used when a close of an output stream fails.

6.344.4.2 `const int decaf::util::logging::ErrorManager::FLUSH_FAILURE`  
[static]

FLUSH\_FAILURE is used when a flush to an output stream fails.

6.344.4.3 `const int decaf::util::logging::ErrorManager::FORMAT_FAILURE`  
[static]

FORMAT\_FAILURE is used when formatting fails for any reason.

6.344.4.4 `const int decaf::util::logging::ErrorManager::GENERIC_FAILURE`  
[static]

GENERIC\_FAILURE is used for failure that don't fit into one of the other categories.

6.344.4.5 `const int decaf::util::logging::ErrorManager::OPEN_FAILURE`  
[static]

OPEN\_FAILURE is used when an open of an output stream fails.

6.344.4.6 `const int decaf::util::logging::ErrorManager::WRITE_FAILURE`  
[static]

WRITE\_FAILURE is used when a write to an output stream fails.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/ErrorManager.h`

### 6.345 decaf::lang::Exception Class Reference

```
#include <src/main/decaf/lang/Exception.h>
```

Inheritance diagram for decaf::lang::Exception:

## Public Member Functions

- **Exception** () throw ()  
*Default Constructor.*
- **Exception** (const **Exception** &ex) throw ()  
*Copy Constructor.*
- **Exception** (const std::exception \***cause**) throw ()  
*Constructor.*
- **Exception** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **Exception** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual ~**Exception** () throw ()
- virtual std::string **getMessage** () const  
*Gets the message for this exception.*
- virtual const std::exception \* **getCause** () const  
*Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual void **initCause** (const std::exception \***cause**)  
*Initializes the contained cause exception with the one given.*
- virtual const char \* **what** () const throw ()  
*Implement method from std::exception.*
- virtual void **setMessage** (const char \*msg,...)  
*Sets the cause for this exception.*
- virtual void **setMark** (const char \*file, const int lineNumber)  
*Adds a file/line number to the stack trace.*
- virtual **Exception** \* **clone** () const  
*Clones this exception.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **printStackTrace** () const  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const  
*Gets the stack trace as one contiguous string.*
- **Exception** & **operator=** (const **Exception** &ex)  
*Assignment operator.*



## Protected Member Functions

- virtual void **setStackTrace** (const std::vector< std::pair< std::string, int > > &trace)
- virtual void **buildMessage** (const char \*format, va\_list &vargs)

## Protected Attributes

- std::string **message**  
*The cause of this exception.*
- std::exception \* **cause**  
*The **Exception** (p. 1794) that caused this one to be thrown.*
- std::vector< std::pair< std::string, int > > **stackTrace**  
*The stack trace.*

## 6.345.1 Constructor & Destructor Documentation

### 6.345.1.1 decaf::lang::Exception::Exception ( ) throw ()

Default Constructor.

### 6.345.1.2 decaf::lang::Exception::Exception ( const Exception & ex ) throw ()

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) instance to copy.
-----------	--

### 6.345.1.3 decaf::lang::Exception::Exception ( const std::exception \* cause ) throw ()

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

### 6.345.1.4 decaf::lang::Exception::Exception ( const char \* file, const int lineNumber, const char \* msg, ... ) throw ()

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.345.1.5 `decaf::lang::Exception::Exception ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.345.1.6 `virtual decaf::lang::Exception::~~Exception ( ) throw () [virtual]`

**6.345.2 Member Function Documentation**

6.345.2.1 `virtual void decaf::lang::Exception::buildMessage ( const char * format, va_list & vargs ) [protected, virtual]`

Referenced by `decaf::lang::exceptions::NumberFormatException::NumberFormatException()`.

6.345.2.2 `virtual Exception* decaf::lang::Exception::clone ( ) const [virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

Copy of this **Exception** (p. 1794) object

Implements **decaf::lang::Throwable** (p. 3725).

Reimplemented in **activemq::exceptions::ActiveMQException** (p. 330), **activemq::exceptions::BrokerException** (p. 828), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2824), **decaf::io::EOFException** (p. 1791), **decaf::io::InterruptedIOException** (p. 2091), **decaf::io::IOException** (p. 2105), **decaf::io::UnsupportedEncodingException** (p. 3849),

**decaf::io::UTFDataFormatException** (p. 3900), **decaf::lang::exceptions::ClassCastException** (p. 1119), **decaf::lang::exceptions::IllegalArgumentException** (p. 1955), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1957), **decaf::lang::exceptions::IllegalStateException** (p. 1961), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1964), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1970), **decaf::lang::exceptions::InterruptedException** (p. 2089), **decaf::lang::exceptions::InvalidStateException** (p. 2102), **decaf::lang::exceptions::NoSuchElementException** (p. 2781), **decaf::lang::exceptions::NullPointerException** (p. 2786), **decaf::lang::exceptions::NumberFormatException** (p. 2791), **decaf::lang::exceptions::RuntimeException** (p. 3269), **decaf::lang::exceptions::UnsupportedOperationException** (p. 3852), **decaf::net::BindException** (p. 800), **decaf::net::ConnectException** (p. 1232), **decaf::net::HttpRetryException** (p. 1950), **decaf::net::MalformedURLException** (p. 2418), **decaf::net::NoRouteToHostException** (p. 2775), **decaf::net::PortUnreachableException** (p. 2924), **decaf::net::ProtocolException** (p. 3085), **decaf::net::SocketException** (p. 3467), **decaf::net::SocketTimeoutException** (p. 3489), **decaf::net::UnknownHostException** (p. 3844), **decaf::net::UnknownServiceException** (p. 3846), **decaf::net::URISyntaxException** (p. 3883), **decaf::nio::BufferOverflowException** (p. 916), **decaf::nio::BufferUnderflowException** (p. 918), **decaf::nio::InvalidMarkException** (p. 2099), **decaf::nio::ReadOnlyBufferException** (p. 3117), **decaf::security::cert::CertificateEncodingException** (p. 1061), **decaf::security::cert::CertificateException** (p. 1062), **decaf::security::cert::CertificateExpiredException** (p. 1064), **decaf::security::cert::CertificateNotYetValidException** (p. 1066), **decaf::security::cert::CertificateParsingException** (p. 1068), **decaf::security::GeneralSecurityException** (p. 1936), **decaf::security::InvalidKeyException** (p. 2096), **decaf::security::KeyException** (p. 2257), **decaf::security::KeyManagementException** (p. 2260), **decaf::security::NoSuchAlgorithmException** (p. 2778), **decaf::security::NoSuchProviderException** (p. 2783), **decaf::security::SignatureException** (p. 3442), **decaf::util::concurrent::BrokenBarrierException** (p. 823), **decaf::util::concurrent::CancellationException** (p. 1055), **decaf::util::concurrent::ExecutionException** (p. 1831), **decaf::util::concurrent::RejectedExecutionException** (p. 3136), **decaf::util::concurrent::TimeoutException** (p. 3730), **decaf::util::zip::DataFormatException** (p. 1522), and **decaf::util::zip::ZipException** (p. 3993).

**6.345.2.3** `virtual const std::exception* decaf::lang::Exception::getCause ( ) const` `[inline, virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

### Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 3726).

**6.345.2.4** `virtual std::string decaf::lang::Exception::getMessage ( ) const` `[inline, virtual]`

Gets the message for this exception.

### Returns

Text formatted error message

Implements **decaf::lang::Throwable** (p. 3726).

Referenced by `activemq::exceptions::BrokerException::BrokerException()`.

6.345.2.5 `virtual std::vector< std::pair< std::string, int> >  
decaf::lang::Exception::getStackTrace ( ) const [virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

#### Returns

the stack trace.

Implements **decaf::lang::Throwable** (p. 3726).

6.345.2.6 `virtual std::string decaf::lang::Exception::getStackTraceString ( ) const  
[virtual]`

Gets the stack trace as one contiguous string.

#### Returns

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 3727).

6.345.2.7 `virtual void decaf::lang::Exception::initCause ( const std::exception * cause )  
[virtual]`

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

#### Parameters

<i>cause</i>	The exception that was the cause of this one.
--------------	---

Implements **decaf::lang::Throwable** (p. 3727).

6.345.2.8 `Exception& decaf::lang::Exception::operator= ( const Exception & ex )`

Assignment operator.

#### Parameters

<i>ex</i>	const reference to another <b>Exception</b> (p. 1794)
-----------	---

6.345.2.9 virtual void decaf::lang::Exception::printStackTrace ( ) const [virtual]

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 3727).

6.345.2.10 virtual void decaf::lang::Exception::printStackTrace ( std::ostream & *stream* ) const [virtual]

Prints the stack trace to the given output stream.

#### Parameters

<i>stream</i>	the target output stream.
---------------	---------------------------

Implements **decaf::lang::Throwable** (p. 3727).

6.345.2.11 virtual void decaf::lang::Exception::setMark ( const char \* *file*, const int *lineNumber* ) [virtual]

Adds a file/line number to the stack trace.

#### Parameters

<i>file</i>	The name of the file calling this method (use __FILE__).
<i>lineNumber</i>	The line number in the calling file (use __LINE__).

Implements **decaf::lang::Throwable** (p. 3728).

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.345.2.12 virtual void decaf::lang::Exception::setMessage ( const char \* *msg*, ... ) [virtual]

Sets the cause for this exception.

#### Parameters

<i>msg</i>	the format string for the msg.
...	params to format into the string

6.345.2.13 virtual void decaf::lang::Exception::setStackTrace ( const std::vector< std::pair< std::string, int > > & *trace* ) [protected, virtual]

6.345.2.14 virtual const char\* decaf::lang::Exception::what ( ) const throw () [inline, virtual]

Implement method from std::exception.

**Returns**

the const char\* of **getMessage()** (p. 1798).

**6.345.3 Field Documentation**

**6.345.3.1** `std::exception* decaf::lang::Exception::cause` [protected]

The **Exception** (p. 1794) that caused this one to be thrown.

**6.345.3.2** `std::string decaf::lang::Exception::message` [protected]

The cause of this exception.

**6.345.3.3** `std::vector< std::pair< std::string, int> > decaf::lang::Exception::stackTrace`  
[protected]

The stack trace.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Exception.h**

**6.346 cms::ExceptionListener Class Reference**

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1801) that is registered with the **Connection** (p. 1232).

```
#include <src/main/cms/ExceptionListener.h>
```

**Public Member Functions**

- virtual **~ExceptionListener()**
- virtual void **onException** (const **cms::CMSException** &ex)=0

*Called when an exception occurs.*

**6.346.1 Detailed Description**

If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1801) that is registered with the **Connection** (p. 1232).

An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

**Since**

1.0

**6.346.2 Constructor & Destructor Documentation**

6.346.2.1 `virtual cms::ExceptionListener::~~ExceptionListener ( ) [inline, virtual]`

**6.346.3 Member Function Documentation**

6.346.3.1 `virtual void cms::ExceptionListener::onException ( const cms::CMSException & ex ) [pure virtual]`

Called when an exception occurs.

Once notified of an exception the caller should no longer use the resource that generated the exception.

**Parameters**

<i>ex</i>	Exception Object that occurred.
-----------	---------------------------------

The documentation for this class was generated from the following file:

- src/main/cms/**ExceptionListener.h**

**6.347 activemq::commands::ExceptionResponse Class Reference**

```
#include <src/main/activemq/commands/ExceptionResponse.h>
```

Inheritance diagram for `activemq::commands::ExceptionResponse`:

**Public Member Functions**

- **ExceptionResponse** ()
- virtual **~ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ExceptionResponse** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataSet** \*value) const  
*Compares the **DataSet** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

### Static Public Attributes

- static const unsigned char **ID\_EXCEPTIONRESPONSE** = 31

### Protected Attributes

- **Pointer**< **BrokerError** > **exception**

## 6.347.1 Constructor & Destructor Documentation

6.347.1.1 **activemq::commands::ExceptionResponse::ExceptionResponse** ( )

6.347.1.2 **virtual activemq::commands::ExceptionResponse::~~ExceptionResponse** ( )  
[virtual]

## 6.347.2 Member Function Documentation

6.347.2.1 **virtual ExceptionResponse\* activemq::commands::ExceptionResponse::cloneDataSet** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 3228).

6.347.2.2 **virtual void activemq::commands::ExceptionResponse::copyDataSet** ( const **DataSet** \* src ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.



**Parameters**

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **activemq::commands::Response** (p. 3229).

**6.347.2.3** `virtual bool activemq::commands::ExceptionResponse::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 3229).

**6.347.2.4** `virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Response** (p. 3230).

**6.347.2.5** `virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ( ) [virtual]`

**6.347.2.6** `virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException ( ) const [virtual]`

**6.347.2.7** `virtual void activemq::commands::ExceptionResponse::setException ( const Pointer< BrokerError > & exception ) [virtual]`

**6.347.2.8** `virtual std::string activemq::commands::ExceptionResponse::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3230).

**6.347.3 Field Documentation**

**6.347.3.1** **Pointer<BrokerError> activemq::commands::ExceptionResponse::exception**  
[protected]

**6.347.3.2** **const unsigned char activemq::commands::ExceptionResponse::ID\_ - EXCEPTIONRESPONSE = 31** [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ExceptionResponse.h**

**6.348 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller Class Reference**

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1804).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller**:

**Public Member Functions**

- **ExceptionResponseMarshaller ()**
- virtual **~ExceptionResponseMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*

6.348

**activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller**

**Class Reference**

1813

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.348.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1804).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.348.2 Constructor & Destructor Documentation

6.348.2.1 **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.348.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.348.3 Member Function Documentation

6.348.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

6.348.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

```
6.348.3.3  virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

```
6.348.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262).

6.348

activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller

## Class Reference

1815

```
6.348.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal1  
    ( OpenWireFormat * wireFormat, commands::DataStructure *  
      dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
    [virtual]
```

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
(p. 3262).

```
6.348.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightMarshal2  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw  
    ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
(p. 3263).

```
6.348.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3264).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ExceptionResponseMarshaller.h**

## 6.349 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1809).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ExceptionRespon
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**:

#### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

## 6.349

**activemq:wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

**Class Reference**

**1817**

*Un-marshall an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.349.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1809).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.349.2 Constructor & Destructor Documentation

6.349.2.1 **activemq:wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.349.2.2 **virtual activemq:wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.349.3 Member Function Documentation

6.349.3.1 **virtual commands::DataStructure\* activemq:wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq:wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3242).

```
6.349.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3242).

```
6.349.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.349.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



6.349

**activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

**Class Reference**

1819

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.349.3.5  virtual int activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

```
6.349.3.6  virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

```
6.349.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ExceptionResponseMarshaller.h**

## 6.350 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1813).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**:

#### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

## 6.350

**activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

**Class Reference**

**1821**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.350.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1813).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.350.2 Constructor & Destructor Documentation

6.350.2.1 **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [**inline**]

6.350.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [**inline**, **virtual**]

### 6.350.3 Member Function Documentation

6.350.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3251).

```
6.350.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

```
6.350.3.3  virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

```
6.350.3.4  virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.350

**activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

### Class Reference

1823

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253).

```
6.350.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253).

```
6.350.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254).

```
6.350.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ExceptionResponseMarshaller.h**

## 6.351 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1817).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**:

#### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

## 6.351

**activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

**Class Reference**

**1825**

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.351.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1817).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.351.2 Constructor & Destructor Documentation

6.351.2.1 **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [**inline**]

6.351.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [**inline**, **virtual**]

### 6.351.3 Member Function Documentation

6.351.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

```
6.351.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

```
6.351.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

```
6.351.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



6.351

**activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

**Class Reference**

1827

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

6.351.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

6.351.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3249).

```
6.351.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ExceptionResponseMarshaller.h**

## 6.352 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1821).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**:

#### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

6.352

**activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

**Class Reference**

1829

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.352.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1821).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.352.2 Constructor & Destructor Documentation

6.352.2.1 **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.352.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.352.3 Member Function Documentation

6.352.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3237).

```
6.352.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

```
6.352.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

```
6.352.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.352

**activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

**Class Reference**

1831

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

```
6.352.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

```
6.352.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

```
6.352.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ExceptionResponseMarshaller.h**

## 6.353 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1825).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**:

#### Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

6.353

**activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

**Class Reference**

**1833**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.353.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1825).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.353.2 Constructor & Destructor Documentation

6.353.2.1 **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::ExceptionResponseMarshaller**  
( ) [inline]

6.353.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::~~ExceptionResponseMarshaller**  
( ) [inline, virtual]

### 6.353.3 Member Function Documentation

6.353.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

```
6.353.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

```
6.353.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.353.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



### 6.353

**activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

#### Class Reference

1835

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.353.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

```
6.353.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

```

6.353.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3259).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ExceptionResponseMarshaller.h**

## 6.354 decaf::util::concurrent::ExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/ExecutionException.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutionException:

#### Public Member Functions

- **ExecutionException** () throw ()  
*Default Constructor.*
- **ExecutionException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ExecutionException** (const ExecutionException &ex) throw ()  
*Copy Constructor.*
- **ExecutionException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ExecutionException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- **ExecutionException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ExecutionException** \* **clone** () const  
*Clones this exception.*
- virtual ~**ExecutionException** () throw ()

### 6.354.1 Constructor & Destructor Documentation

6.354.1.1 decaf::util::concurrent::ExecutionException::ExecutionException ( ) throw ()  
[inline]

Default Constructor.

6.354.1.2 decaf::util::concurrent::ExecutionException::ExecutionException ( const decaf::lang::Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	- An exception that should become this type of Exception
-----------	--

6.354.1.3 decaf::util::concurrent::ExecutionException::ExecutionException ( const ExecutionException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

6.354.1.4 decaf::util::concurrent::ExecutionException::ExecutionException ( const std::exception \* cause ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.354.1.5 `decaf::util::concurrent::ExecutionException::ExecutionException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
...	- The list of primitives that are formatted into the message

6.354.1.6 `decaf::util::concurrent::ExecutionException::ExecutionException ( const char * file,  
const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()  
[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.354.1.7 `virtual decaf::util::concurrent::ExecutionException::~~ExecutionException ( ) throw ()  
[inline, virtual]`

### 6.354.2 Member Function Documentation

6.354.2.1 `virtual ExecutionException* decaf::util::concurrent::ExecutionException::clone ( )const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutionException.h**

## 6.355 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang Runnable** (p. 3264) tasks.

```
#include <src/main/decaf/util/concurrent/Executor.h>
```

Inheritance diagram for decaf::util::concurrent::Executor:

### Public Member Functions

- virtual **~Executor** ()
- virtual void **execute** (Runnable \*command)=0 throw ( decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException )

*Executes the given command at some time in the future.*

### 6.355.1 Detailed Description

An object that executes submitted **decaf.lang Runnable** (p. 3264) tasks.

This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An **Executor** (p. 1831) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread(new RunnableTask()) .start()` for each of a set of tasks, you might use:

```
Executor (p.1831) executor = anExecutor;
executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...
```

However, the **Executor** (p. 1831) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```
class DirectExecutor : public Executor (p.1831) {
public:

    void execute( Runnable* r ) (p.1833) {
        r->run();
    }
}
```

```
}
```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```
class ThreadPerTaskExecutor : public Executor (p.1831) {
public:
    std::vector<Thread*gt; threads;

    void execute( Runnable* r ) (p.1833) {
        threads.push_back( new Thread( r ) );
        threads.rbegin()->start();
    }
}
```

The **Executor** (p.1831) implementations provided in this package implement **decaf.util.concurrent.ExecutorService** (p.1833), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p.??) class provides an extensible thread pool implementation. The **decaf.util.concurrentExecutor** (p.??) class provides convenient factory methods for these Executors.

#### Since

1.0

### 6.355.2 Constructor & Destructor Documentation

6.355.2.1 `virtual decaf::util::concurrent::Executor::~~Executor ( ) [inline, virtual]`

### 6.355.3 Member Function Documentation

6.355.3.1 `virtual void decaf::util::concurrent::Executor::execute ( Runnable* command ) throw ( decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException ) [pure virtual]`

Executes the given command at some time in the future.

The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the **Executor** (p.1831) implementation.

#### Parameters

<i>command</i>	the runnable task
----------------	-------------------

#### Exceptions

<b>RejectedExecutionException</b> (p. 3134)	if this task cannot be accepted for execution.
<b>NullPointerException</b>	if command is null

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Executor.h**

## 6.356 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p. 1831) that provides methods to manage termination and methods that can produce a **Future** (p. 1929) for tracking progress of one or more asynchronous tasks.

```
#include <src/main/decaf/util/concurrent/ExecutorService.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutorService:

### Public Member Functions

- virtual **~ExecutorService** ()
- bool **awaitTermination** (long long timeout, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::InterruptedException )

*Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.*

### 6.356.1 Detailed Description

An **Executor** (p. 1831) that provides methods to manage termination and methods that can produce a **Future** (p. 1929) for tracking progress of one or more asynchronous tasks.

An **ExecutorService** (p. 1833) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p. 1833). The shutdown() method will allow previously submitted tasks to execute before terminating, while the shutdownNow() method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused **ExecutorService** (p. 1833) should be shut down to allow reclamation of its resources.

Method submit extends base method **Executor.execute** (p. 1833)(**decaf.lang Runnable** (p. 3264)) by creating and returning a **Future** (p. 1929) that can be used to cancel execution and/or wait for completion. Methods invokeAny and invokeAll perform the most

commonly useful forms of bulk execution, executing a collection of tasks and then waiting for at least one, or all, to complete. (Class `ExecutorCompletionService` can be used to write customized variants of these methods.)

The `Executors` class provides factory methods for the executor services provided in this package.

#### Since

1.0

### 6.356.2 Constructor & Destructor Documentation

6.356.2.1 `virtual decaf::util::concurrent::ExecutorService::~~ExecutorService ( )`  
`[inline, virtual]`

### 6.356.3 Member Function Documentation

6.356.3.1 `bool decaf::util::concurrent::ExecutorService::awaitTermination`  
`( long long timeout, const TimeUnit & unit ) throw (`  
`decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

#### Parameters

<i>timeout</i>	The amount of time to wait before timing out the Wait operation.
<i>unit</i>	The Units that comprise the timeout value.

#### Returns

true if the executor terminated before the given timeout value elapsed.

#### Exceptions

<i>InterruptedException</i>	- if interrupted while waiting.
-----------------------------	---------------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ExecutorService.h`

## 6.357 activemq::transport::failover::FailoverTransport Class Reference

```
#include <src/main/activemq/transport/failover/FailoverTransport.h>
```



Inheritance diagram for activemq::transport::failover::FailoverTransport:

## Public Member Functions

- **FailoverTransport** ()
- virtual **~FailoverTransport** ()
- void **reconnect** ()
  - Indicates that the **Transport** (p. 3819) needs to reconnect to another URI in its list.*
- void **add** (const std::string &uri)
  - Adds a New URI to the List of URIs this transport can Connect to.*
- virtual void **addURI** (const **List**< **URI** > &uris)
  - Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3819) is a composite of.*
- virtual void **removeURI** (const **List**< **URI** > &uris)
  - Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3819) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3819) should result in that **Transport** (p. 3819) being disposed of.*
- virtual void **start** () throw ( decaf::io::IOException )
  - Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **stop** () throw ( decaf::io::IOException )
  - Stop the **Transport** (p. 3819).*
- virtual void **close** () throw ( decaf::io::IOException )
  - Stops the polling thread and closes the streams.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )
  - Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )
  - Sends the given command to the broker and then waits for the response.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )
  - Sends the given command to the broker and then waits for the response.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP\_UNUSED)
  - Sets the WireFormat instance to use.*
- virtual void **setTransportListener** (**TransportListener** \*listener)
  - Sets the observer of asynchronous events from this transport.*
- virtual **TransportListener** \* **getTransportListener** () const
  - Gets the observer of asynchronous exceptions from this transport.*

- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3819) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3819) been shutdown and no longer usable.*
- bool **isInitialized** () const  
*Returns true if the **Transport** (p. 3819) has been initialized by a BrokerInfo command.*
- void **setInitialized** (bool value)  
*Sets the initialized state of this **Transport** (p. 3819) to true.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeid)  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual std::string **getRemoteAddress** () const
- virtual bool **isPending** () const
- virtual bool **iterate** ()  
*Performs the actual Reconnect operation for the **FailoverTransport** (p. 1835), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.*
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw ( decaf::io::IOException )  
*reconnect to another location*
- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const
- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)
- int **getStartupMaxReconnectAttempts** () const
- void **setStartupMaxReconnectAttempts** (int value)
- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const
- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const

- void **setTrackMessages** (bool value)
- bool **isTrackTransactionProducers** () const
- void **setTrackTransactionProducers** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)
- void **setConnectionInterruptProcessingComplete** (const **Pointer**< **commands::ConnectionId** > &connectionId)

### Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport) throw ( **decaf::io::IOException** )  
*Given a **Transport** (p. 3819) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.*
- void **handleTransportFailure** (const **decaf::lang::Exception** &error) throw ( **decaf::lang::Exception** )  
*Called when this class' **TransportListener** (p. 3836) is notified of a Failure.*

### Friends

- class **FailoverTransportListener**

## 6.357.1 Constructor & Destructor Documentation

6.357.1.1 **activemq::transport::failover::FailoverTransport::FailoverTransport ( )**

6.357.1.2 **virtual activemq::transport::failover::FailoverTransport::~~FailoverTransport ( )**  
 [virtual]

## 6.357.2 Member Function Documentation

6.357.2.1 **void activemq::transport::failover::FailoverTransport::add ( const std::string & uri )**

Adds a New URI to the List of URIs this transport can Connect to.

### Parameters

<i>uri</i>	A String version of a URI to add to the URIs to failover to.
------------	--

6.357.2.2 **virtual void activemq::transport::failover::FailoverTransport::addURI ( const List< URI > & uris )** [virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3819) is a composite of.

**Parameters**

<i>uris</i>	The new URIs to add to the set this composite maintains.
-------------	--

Implements **activemq::transport::CompositeTransport** (p. 1197).

6.357.2.3 `virtual void activemq::transport::failover::FailoverTransport::close ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

**Exceptions**

<i>IOException</i>	if errors occur.
--------------------	------------------

Implements **decaf::io::Closeable** (p. 1121).

6.357.2.4 `long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier ( ) const [inline]`

6.357.2.5 `int activemq::transport::failover::FailoverTransport::getBackupPoolSize ( ) const [inline]`

6.357.2.6 `long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay ( ) const [inline]`

6.357.2.7 `int activemq::transport::failover::FailoverTransport::getMaxCacheSize ( ) const [inline]`

6.357.2.8 `int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts ( ) const [inline]`

6.357.2.9 `long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay ( ) const [inline]`

6.357.2.10 `long long activemq::transport::failover::FailoverTransport::getReconnectDelay ( ) const [inline]`

6.357.2.11 `virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress ( ) const [virtual]`

**Returns**

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3821).

6.357.2.12 `int activemq::transport::failover::FailoverTransport::getStartupMaxReconnectAttempts ( ) const [inline]`

6.357.2.13 `long long activemq::transport::failover::FailoverTransport::getTimeout ( ) const [inline]`

6.357.2.14 `virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener ( ) const [virtual]`

Gets the observer of asynchronous exceptions from this transport.

#### Returns

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3821).

6.357.2.15 `void activemq::transport::failover::FailoverTransport::handleTransportFailure ( const decaf::lang::Exception & error ) throw ( decaf::lang::Exception ) [protected]`

Called when this class' **TransportListener** (p. 3836) is notified of a Failure.

#### Parameters

<i>error</i>	- The CMS Exception that was thrown.
--------------	--------------------------------------

#### Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

6.357.2.16 `bool activemq::transport::failover::FailoverTransport::isBackup ( ) const [inline]`

6.357.2.17 `virtual bool activemq::transport::failover::FailoverTransport::isClosed ( ) const [inline, virtual]`

Has the **Transport** (p. 3819) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3819)

Implements **activemq::transport::Transport** (p. 3821).

6.357.2.18 `virtual bool activemq::transport::failover::FailoverTransport::isConnected ( ) const`  
`[inline, virtual]`

Is the **Transport** (p. 3819) Connected to its Broker.

#### Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3821).

6.357.2.19 `virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant ( )`  
`const [inline, virtual]`

Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns

true if the **Transport** (p. 3819) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3822).

6.357.2.20 `bool activemq::transport::failover::FailoverTransport::isInitialized ( ) const`  
`[inline]`

Returns true if the **Transport** (p. 3819) has been initialized by a BrokerInfo command.

#### Returns

true if the **Transport** (p. 3819) has been initialized by a BrokerInfo command.

6.357.2.21 `virtual bool activemq::transport::failover::FailoverTransport::isPending ( ) const`  
`[virtual]`

#### Returns

true if there is a need for the iterate method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 1194).

6.357.2.22 `bool activemq::transport::failover::FailoverTransport::isRandomize ( ) const`  
`[inline]`

6.357.2.23 `bool activemq::transport::failover::FailoverTransport::isTrackMessages ( ) const`  
`[inline]`

6.357.2.24 `bool activemq::transport::failover::FailoverTransport::isTrackTransactionProducers ( ) const [inline]`

6.357.2.25 `bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff ( ) const [inline]`

6.357.2.26 `virtual bool activemq::transport::failover::FailoverTransport::iterate ( ) [virtual]`

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1835), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

#### Returns

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 3679).

6.357.2.27 `virtual Transport* activemq::transport::failover::FailoverTransport::narrow ( const std::type_info & typeId ) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.

#### Parameters

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

#### Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3822).

References `activemq::transport::Transport::narrow()`.

6.357.2.28 `virtual void activemq::transport::failover::FailoverTransport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

**Exceptions**

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3822).

6.357.2.29 `void activemq::transport::failover::FailoverTransport::reconnect ( )`

Indicates that the **Transport** (p. 3819) needs to reconnect to another URI in its list.

6.357.2.30 `virtual void activemq::transport::failover::FailoverTransport::reconnect ( const decaf::net::URI & uri ) throw ( decaf::io::IOException ) [virtual]`

reconnect to another location

**Parameters**

<i>uri</i>	
------------	--

**Exceptions**

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

Implements **activemq::transport::Transport** (p. 3823).

6.357.2.31 `virtual void activemq::transport::failover::FailoverTransport::removeURI ( const List< URI > & uris ) [virtual]`

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3819) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3819) should result in that **Transport** (p. 3819) being disposed of.

**Parameters**

<i>uris</i>	The new URIs to remove to the set this composite maintains.
-------------	---

Implements **activemq::transport::CompositeTransport** (p. 1198).

6.357.2.32 `virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends the given command to the broker and then waits for the response.



**Parameters**

<i>command</i>	the command to be sent.
----------------	-------------------------

**Returns**

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3823).

```
6.357.2.33 virtual Pointer<Response> ac-
tivemq::transport::failover::FailoverTransport::request (
const Pointer< Command > & command, unsigned
int timeout ) throw ( decaf::io::IOException, de-
caf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Sends the given command to the broker and then waits for the response.

**Parameters**

<i>command</i>	- The command to be sent.
<i>timeout</i>	- The time to wait for this response.

**Returns**

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3824).

```
6.357.2.34 void activemq::transport::failover::FailoverTransport::restoreTransport ( const
Pointer< Transport > & transport ) throw ( decaf::io::IOException )
[protected]
```

Given a **Transport** (p. 3819) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.

**Parameters**

<i>transport</i>	The new <b>Transport</b> (p. 3819) connected to the Broker.
------------------	---

**Exceptions**

<i>IOException</i>	if an errors occurs while restoring the old state.
--------------------	--

- 6.357.2.35 `void activemq::transport::failover::FailoverTransport::setBackOffMultiplier ( long long value ) [inline]`
- 6.357.2.36 `void activemq::transport::failover::FailoverTransport::setBackup ( bool value ) [inline]`
- 6.357.2.37 `void activemq::transport::failover::FailoverTransport::setBackupPoolSize ( int value ) [inline]`
- 6.357.2.38 `void activemq::transport::failover::FailoverTransport::setConnectionInterruptProcessingComplete ( const Pointer< commands::ConnectionId > & connectionId )`
- 6.357.2.39 `void activemq::transport::failover::FailoverTransport::setInitialized ( bool value ) [inline]`

Sets the initialized state of this **Transport** (p. 3819) to true.

**Parameters**

<i>value</i>	- true if this <b>Transport</b> (p. 3819) has been initialized.
--------------	---

- 6.357.2.40 `void activemq::transport::failover::FailoverTransport::setInitialReconnectDelay ( long long value ) [inline]`
- 6.357.2.41 `void activemq::transport::failover::FailoverTransport::setMaxCacheSize ( int value ) [inline]`
- 6.357.2.42 `void activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts ( int value ) [inline]`
- 6.357.2.43 `void activemq::transport::failover::FailoverTransport::setMaxReconnectDelay ( long long value ) [inline]`
- 6.357.2.44 `void activemq::transport::failover::FailoverTransport::setRandomize ( bool value ) [inline]`
- 6.357.2.45 `void activemq::transport::failover::FailoverTransport::setReconnectDelay ( long long value ) [inline]`

6.357.2.46 `void activemq::transport::failover::FailoverTransport::setStartupMaxReconnectAttempts ( int value ) [inline]`

6.357.2.47 `void activemq::transport::failover::FailoverTransport::setTimeout ( long long value ) [inline]`

6.357.2.48 `void activemq::transport::failover::FailoverTransport::setTrackMessages ( bool value ) [inline]`

6.357.2.49 `void activemq::transport::failover::FailoverTransport::setTrackTransactionProducers ( bool value ) [inline]`

6.357.2.50 `virtual void activemq::transport::failover::FailoverTransport::setTransportListener ( TransportListener * listener ) [virtual]`

Sets the observer of asynchronous events from this transport.

#### Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p.3824).

6.357.2.51 `void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff ( bool value ) [inline]`

6.357.2.52 `virtual void activemq::transport::failover::FailoverTransport::setWireFormat ( const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED ) [inline, virtual]`

Sets the WireFormat instance to use.

#### Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

6.357.2.53 `virtual void activemq::transport::failover::FailoverTransport::start ( ) throw ( decaf::io::IOException ) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

#### Exceptions

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Implements **activemq::transport::Transport** (p. 3825).

6.357.2.54 `virtual void activemq::transport::failover::FailoverTransport::stop ( ) throw ( decaf::io::IOException )` [virtual]

Stop the **Transport** (p. 3819).

#### Exceptions

<i>IOException</i>	if an error occurs while stopping the <b>Transport</b> (p. 3819).
--------------------	---

Implements **activemq::transport::Transport** (p. 3825).

### 6.357.3 Friends And Related Function Documentation

6.357.3.1 `friend class FailoverTransportListener` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransport.h`

## 6.358 **activemq::transport::failover::FailoverTransportFactory Class Reference**

Creates an instance of a **FailoverTransport** (p. 1835).

```
#include <src/main/activemq/transport/failover/FailoverTransportFactory.h>
```

Inheritance diagram for **activemq::transport::failover::FailoverTransportFactory**:

#### Public Member Functions

- `virtual ~FailoverTransportFactory ()`
- `virtual Pointer< Transport > create (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`  
*Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.*
- `virtual Pointer< Transport > createComposite (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`  
*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

## 6.358 activemq::transport::failover::FailoverTransportFactory Class Reference

### Protected Member Functions

- virtual **Pointer**< **Transport** > **doCreateComposite** (const **decaf::net::URI** &location, const **decaf::util::Properties** &properties) throw ( exceptions::ActiveMQException )

*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

### 6.358.1 Detailed Description

Creates an instance of a **FailoverTransport** (p. 1835).

Since

3.0

### 6.358.2 Constructor & Destructor Documentation

6.358.2.1 virtual **activemq::transport::failover::FailoverTransportFactory::~FailoverTransportFactory**  
( ) [inline, virtual]

### 6.358.3 Member Function Documentation

6.358.3.1 virtual **Pointer**<**Transport**> **activemq::transport::failover::FailoverTransportFactory::create** ( const **decaf::net::URI** &location ) throw ( exceptions::ActiveMQException )  
[virtual]

Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 3826).

6.358.3.2 virtual **Pointer**<**Transport**> **activemq::transport::failover::FailoverTransportFactory::createComposite** ( const **decaf::net::URI** &location ) throw ( exceptions::ActiveMQException )  
[virtual]

Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.

**Parameters**

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

**Exceptions**

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 3827).

```
6.358.3.3 virtual Pointer<Transport> ac-
    tivemq::transport::failover::FailoverTransportFactory::doCreateComposite ( const
    decaf::net::URI & location, const decaf::util::Properties & properties ) throw (
    exceptions::ActiveMQException ) [protected, virtual]
```

Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.

**Parameters**

<i>location</i>	- URI location to connect to.
<i>properties</i>	- Properties to apply to the transport.

**Returns**

Pointer to a new **FailoverTransport** (p. 1835) instance.

**Exceptions**

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportFactory.h**

## 6.359 **activemq::transport::failover::FailoverTransportListener** Class Reference

Utility class used by the **Transport** (p. 3819) to perform the work of responding to events from the active **Transport** (p. 3819).

```
#include <src/main/activemq/transport/failover/FailoverTransportListener.h>
```

Inheritance diagram for **activemq::transport::failover::FailoverTransportListener**:

## Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** \*parent)
- virtual ~**FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **transportInterrupted** ()  
*The transport has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
*The transport has resumed after an interruption.*

### 6.359.1 Detailed Description

Utility class used by the **Transport** (p. 3819) to perform the work of responding to events from the active **Transport** (p. 3819).

#### Since

3.0

### 6.359.2 Constructor & Destructor Documentation

6.359.2.1 **activemq::transport::failover::FailoverTransportListener::FailoverTransportListener** (**FailoverTransport** \* parent )

6.359.2.2 virtual **activemq::transport::failover::FailoverTransportListener::~~FailoverTransportListener** ( ) [virtual]

### 6.359.3 Member Function Documentation

6.359.3.1 virtual void **activemq::transport::failover::FailoverTransportListener::onCommand** (const **Pointer**< **Command** > & command ) [virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3819) deletes the command upon receipt.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements **activemq::transport::TransportListener** (p. 3836).

6.359.3.2 `virtual void activemq::transport::failover::FailoverTransportListener::onException (const decaf::lang::Exception & ex ) [virtual]`

Event handler for an exception from a command transport.

#### Parameters

<code>ex</code>	The exception.
-----------------	----------------

Implements `activemq::transport::TransportListener` (p. 3837).

6.359.3.3 `virtual void activemq::transport::failover::FailoverTransportListener::transportInterrupted ( ) [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 3837).

6.359.3.4 `virtual void activemq::transport::failover::FailoverTransportListener::transportResumed ( ) [virtual]`

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 3837).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportListener.h`

## 6.360 decaf::io::FileDescriptor Class Reference

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

```
#include <src/main/decaf/io/FileDescriptor.h>
```

Inheritance diagram for `decaf::io::FileDescriptor`:

#### Public Member Functions

- `FileDescriptor ()`
- `virtual ~FileDescriptor ()`
- `void sync ()`  
*Force any/all buffered data for this **FileDescriptor** (p. 1850) to be flushed to the underlying device.*
- `bool valid ()`  
*Indicates whether the File Descriptor is valid.*



### Static Public Attributes

- static **FileDescriptor** **in**  
*A handle to the standard input stream.*
- static **FileDescriptor** **out**  
*A handle to the standard output stream.*
- static **FileDescriptor** **err**  
*A handle to the standard error stream.*

### Protected Member Functions

- **FileDescriptor** (long value, bool **readonly**)

### Protected Attributes

- long **descriptor**
- bool **readonly**

#### 6.360.1 Detailed Description

This class servers as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.

#### Since

1.0

#### 6.360.2 Constructor & Destructor Documentation

6.360.2.1 `decaf::io::FileDescriptor::FileDescriptor ( long value, bool readonly )`  
[protected]

6.360.2.2 `decaf::io::FileDescriptor::FileDescriptor ( )`

6.360.2.3 `virtual decaf::io::FileDescriptor::~~FileDescriptor ( )` [virtual]

#### 6.360.3 Member Function Documentation

6.360.3.1 `void decaf::io::FileDescriptor::sync ( )`

Force any/all buffered data for this **FileDescriptor** (p. 1850) to be flushed to the underlying device.

This method blocks until all data is flushed to the underlying device and is used to place the device into a known state. In the case of data that is buffered at a higher level such as a **BufferedOutputStream** (p. 899) the stream must first be flushed before this method can force the data to be sent to the output device.

### 6.360.3.2 `bool decaf::io::FileDescriptor::valid ( )`

Indicates whether the File Descriptor is valid.

#### Returns

true for a valid descriptor such as open socket or file, false otherwise.

## 6.360.4 Field Documentation

### 6.360.4.1 `long decaf::io::FileDescriptor::descriptor` `[protected]`

### 6.360.4.2 `FileDescriptor decaf::io::FileDescriptor::err` `[static]`

A handle to the standard error stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as `System::err`.

### 6.360.4.3 `FileDescriptor decaf::io::FileDescriptor::in` `[static]`

A handle to the standard input stream.

Usually, this file descriptor is not used directly, but rather via the input stream known as `System::in`.

### 6.360.4.4 `FileDescriptor decaf::io::FileDescriptor::out` `[static]`

A handle to the standard output stream.

Usually, this file descriptor is not used directly, but rather via the output stream known as `System::out`.

### 6.360.4.5 `bool decaf::io::FileDescriptor::readonly` `[protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FileDescriptor.h`

## 6.361 `decaf::util::logging::Filter` Class Reference

A **Filter** (p. 1853) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

```
#include <src/main/decaf/util/logging/Filter.h>
```

## Public Member Functions

- virtual `~Filter()`
- virtual bool `isLoggable (const LogRecord &record) const` =0

*Check if a given log record should be published.*

### 6.361.1 Detailed Description

A **Filter** (p. 1853) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

Each **Logger** (p. 2345) and each **Handler** (p. 1941) can have a filter associated with it. The **Logger** (p. 2345) or **Handler** (p. 1941) will call the `isLoggable` method to check if a given **LogRecord** (p. 2370) should be published. If `isLoggable` returns false, the **LogRecord** (p. 2370) will be discarded.

### 6.361.2 Constructor & Destructor Documentation

6.361.2.1 virtual `decaf::util::logging::Filter::~Filter ( )` [`inline`, `virtual`]

### 6.361.3 Member Function Documentation

6.361.3.1 virtual bool `decaf::util::logging::Filter::isLoggable ( const LogRecord & record )`  
`const` [`pure virtual`]

Check if a given log record should be published.

#### Parameters

<i>record</i>	the <b>LogRecord</b> (p. 2370) to check.
---------------	--

#### Returns

true if the record is loggable.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Filter.h`

## 6.362 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p. 1854) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

```
#include <src/main/decaf/io/FilterInputStream.h>
```

Inheritance diagram for `decaf::io::FilterInputStream`:

## Public Member Functions

- **FilterInputStream** (**InputStream** \***inputStream**, bool **own**=false)

*Constructor to create a wrapped **InputStream** (p. 2002).*

- virtual ~**FilterInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

### Returns

*the number of bytes available on this input stream.*

### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.*

*The default implementation of this method does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedC )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

### Parameters

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------

### Returns

*total bytes skipped*

### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

<b>UnsupportedOperationException</b>	<i>if the concrete stream class does not support skipping bytes.</i>
--------------------------------------	--

- virtual void **mark** (int readLimit)

*Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the*

same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

readLimit	The max bytes read before marked position is invalid.
-----------	---

- virtual void **reset** () throw ( decaf::io::IOException )

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns

true if this stream instance supports marks

## Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArray** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual bool **isClosed** () const

## Protected Attributes

- **InputStream** \* **inputStream**
- bool **own**
- volatile bool **closed**

### 6.362.1 Detailed Description

A **FilterInputStream** (p. 1854) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

The class **FilterInputStream** (p. 1854) itself simply overrides all methods of **InputStream** (p. 2002) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p. 1854) may further override some of these methods and may also provide additional methods and fields.

### 6.362.2 Constructor & Destructor Documentation

6.362.2.1 `decaf::io::FilterInputStream::FilterInputStream ( InputStream * inputStream, bool own = false )`

Constructor to create a wrapped **InputStream** (p. 2002).

#### Parameters

<i>inputStream</i>	The stream to wrap and filter.
<i>own</i>	Indicates if we own the stream object, defaults to false.

6.362.2.2 `virtual decaf::io::FilterInputStream::~FilterInputStream ( )` [virtual]

### 6.362.3 Member Function Documentation

6.362.3.1 `virtual int decaf::io::FilterInputStream::available ( ) const throw ( decaf::io::IOException )` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2004).

Reimplemented in **decaf::io::BufferedInputStream** (p. 896), **decaf::io::PushbackInputStream** (p. 3089), and **decaf::util::zip::InflaterInputStream** (p. 1998).

**6.362.3.2** `virtual void decaf::io::FilterInputStream::close ( ) throw ( decaf::io::IOException )`  
[virtual]

Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2004).

Reimplemented in **decaf::io::BufferedInputStream** (p. 897), and **decaf::util::zip::InflaterInputStream** (p. 1998).

**6.362.3.3** `virtual int decaf::io::FilterInputStream::doReadArray ( unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2005).

**6.362.3.4** `virtual int decaf::io::FilterInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Reimplemented from **decaf::io::InputStream** (p. 2005).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2359), **decaf::io::BufferedInputStream** (p. 897), **decaf::io::PushbackInputStream** (p. 3090), **decaf::util::zip::CheckedInputStream** (p. 1111), and **decaf::util::zip::InflaterInputStream** (p. 1999).

**6.362.3.5** `virtual int decaf::io::FilterInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Implements **decaf::io::InputStream** (p. 2005).

Reimplemented in **activemq::io::LoggingInputStream** (p. 2359), **decaf::io::BufferedInputStream** (p. 897), **decaf::io::PushbackInputStream** (p. 3090), **decaf::util::zip::CheckedInputStream**

(p. 1111), and **decaf::util::zip::InflaterInputStream** (p. 1999).

**6.362.3.6** `virtual bool decaf::io::FilterInputStream::isClosed ( ) const` [protected, virtual]

#### Returns

true if this stream has been closed.

**6.362.3.7** `virtual void decaf::io::FilterInputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::InputStream** (p. 2006).

Reimplemented in **decaf::io::BufferedInputStream** (p. 897), **decaf::io::PushbackInputStream** (p. 3090), and **decaf::util::zip::InflaterInputStream** (p. 1999).

**6.362.3.8** `virtual bool decaf::io::FilterInputStream::markSupported ( ) const` [virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::InputStream** (p. 2006).

Reimplemented in **decaf::io::BufferedInputStream** (p. 897), **decaf::io::PushbackInputStream** (p. 3090), and **decaf::util::zip::InflaterInputStream** (p. 2000).



6.362.3.9 virtual void decaf::io::FilterInputStream::reset ( ) throw ( decaf::io::IOException )  
[virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2009).

Reimplemented in **decaf::io::BufferedInputStream** (p. 898), **decaf::io::PushbackInputStream** (p. 3091), and **decaf::util::zip::InflaterInputStream** (p. 2000).

6.362.3.10 virtual long long decaf::io::FilterInputStream::skip ( long long num ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

**Returns**

total bytes skipped

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<b><i>UnsupportedOperationException</i></b>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2010).

Reimplemented in **decaf::io::BufferedInputStream** (p. 898), **decaf::io::PushbackInputStream** (p. 3091), **decaf::util::zip::CheckedInputStream** (p. 1112), and **decaf::util::zip::InflaterInputStream** (p. 2001).

**6.362.4 Field Documentation**

**6.362.4.1** `volatile bool decaf::io::FilterInputStream::closed` [protected]

**6.362.4.2** `InputStream* decaf::io::FilterInputStream::inputStream`  
[protected]

**6.362.4.3** `bool decaf::io::FilterInputStream::own` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterInputStream.h`

**6.363 decaf::io::FilterOutputStream Class Reference**

This class is the superclass of all classes that filter output streams.

```
#include <src/main/decaf/io/FilterOutputStream.h>
```

Inheritance diagram for `decaf::io::FilterOutputStream`:

**Public Member Functions**

- **FilterOutputStream** (`OutputStream *outputStream`, `bool own=false`)  
*Constructor, creates a wrapped output stream.*
- virtual **~FilterOutputStream** ()
- virtual void **flush** () throw ( `decaf::io::IOException` )  
*Flushes this stream by writing any buffered output to the underlying stream.*

**Exceptions**

<b>IOException</b> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------

The default implementation of this method does nothing.

- virtual void **close** () throw ( decaf::io::IOException )

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

**Exceptions**

<b>IOException</b> (p. 2103)	if an error occurs while closing.
------------------------------	-----------------------------------

The default implementation of this method does nothing.

- virtual std::string **toString** () const

Output a String representation of this object.

The default version of this method just prints the Class Name.

**Returns**

a string representation of the object.

**Protected Member Functions**

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual bool **isClosed** () const

**Protected Attributes**

- **OutputStream \* outputStream**
- bool **own**
- volatile bool **closed**

**6.363.1 Detailed Description**

This class is the superclass of all classes that filter output streams.

These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1861) itself simply overrides all methods of **OutputStream** (p. 2856) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1861) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 2002) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataOutputStream (p. 1546) os = new DataOutputStream (p. 1546)( new Output-Stream() (p. 2858), true )
```

### 6.363.2 Constructor & Destructor Documentation

6.363.2.1 **decaf::io::FilterOutputStream::FilterOutputStream ( OutputStream \* outputStream, bool own = false )**

Constructor, creates a wrapped output stream.

#### Parameters

<i>output-Stream</i>	the <b>OutputStream</b> (p. 2856) to wrap
<i>own</i>	If true, this object will control the lifetime of the output stream that it encapsulates.

6.363.2.2 **virtual decaf::io::FilterOutputStream::~FilterOutputStream ( )** [virtual]

### 6.363.3 Member Function Documentation

6.363.3.1 **virtual void decaf::io::FilterOutputStream::close ( )** throw ( **decaf::io::IOException** ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The close method of **FilterOutputStream** (p. 1861) calls its flush method, and then calls the close method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2858).

Reimplemented in **decaf::util::zip::DeflaterOutputStream** (p. 1685).

6.363.3.2 `virtual void decaf::io::FilterOutputStream::doWriteArray ( const unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented from **decaf::io::OutputStream** (p. 2858).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 901).

6.363.3.3 `virtual void decaf::io::FilterOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented from **decaf::io::OutputStream** (p. 2859).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 2360), **decaf::io::BufferedOutputStream** (p. 901), **decaf::io::DataOutputStream** (p. 1548), **decaf::util::zip::CheckedOutputStream** (p. 1114), and **decaf::util::zip::DeflaterOutputStream** (p. 1685).

6.363.3.4 `virtual void decaf::io::FilterOutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )` `[protected, virtual]`

Implements **decaf::io::OutputStream** (p. 2859).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 2360), **decaf::io::BufferedOutputStream** (p. 901), **decaf::io::DataOutputStream** (p. 1549), **decaf::util::zip::CheckedOutputStream** (p. 1114), and **decaf::util::zip::DeflaterOutputStream** (p. 1685).

6.363.3.5 `virtual void decaf::io::FilterOutputStream::flush ( ) throw ( decaf::io::IOException )` `[virtual]`

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The flush method of **FilterOutputStream** (p. 1861) calls the flush method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2859).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 901).

6.363.3.6 `virtual bool decaf::io::FilterOutputStream::isClosed ( ) const` [protected, virtual]

#### Returns

true if this stream has been closed.

6.363.3.7 `virtual std::string decaf::io::FilterOutputStream::toString ( ) const` [virtual]

Output a String representation of this object.

The default version of this method just prints the Class Name.

#### Returns

a string representation of the object.

The `toString` method of **FilterOutputStream** (p. 1861) calls the `toString` method of its underlying output stream.

Reimplemented from **decaf::io::OutputStream** (p. 2860).

### 6.363.4 Field Documentation

6.363.4.1 `volatile bool decaf::io::FilterOutputStream::closed` [protected]

6.363.4.2 `OutputStream* decaf::io::FilterOutputStream::outputStream` [protected]

6.363.4.3 `bool decaf::io::FilterOutputStream::own` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterOutputStream.h`

## 6.364 decaf::lang::Float Class Reference

```
#include <src/main/decaf/lang/Float.h>
```

Inheritance diagram for `decaf::lang::Float`:

#### Public Member Functions

- **Float** (float value)
- **Float** (double value)

- **Float** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const  
*Compares this **Float** (p. 1865) instance with another.*
- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Float** &f) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const float &f) const  
*Compares this **Float** (p. 1865) instance with another.*
- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const float &f) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*
- bool **isInfinite** () const
- bool **isNaN** () const

### Static Public Member Functions

- static int **compare** (float f1, float f2)  
*Compares the two specified double values.*
- static int **floatToIntBits** (float value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.*
- static int **floatToRawIntBits** (float value)  
*Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.*

- static float **intBitsToFloat** (int bits)  
*Returns the float value corresponding to a given bit representation.*
- static bool **isInfinite** (float value)
- static bool **isNaN** (float value)
- static float **parseFloat** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1865).*
- static std::string **toHexString** (float value)  
*Returns a hexadecimal string representation of the float argument.*
- static std::string **toString** (float value)  
*Returns a string representation of the float argument.*
- static **Float valueOf** (float value)  
*Returns a **Float** (p. 1865) instance representing the specified float value.*
- static **Float valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Returns a **Float** (p. 1865) instance that wraps a primitive float which is parsed from the string value passed.*

### Static Public Attributes

- static const int **SIZE** = 32  
*The size in bits of the primitive int type.*
- static const float **MAX\_VALUE**  
*The maximum value that the primitive type can hold.*
- static const float **MIN\_VALUE**  
*The minimum value that the primitive type can hold.*
- static const float **NaN**  
*Constant for the Not a **Number** (p. 2786) Value.*
- static const float **POSITIVE\_INFINITY**  
*Constant for Positive Infinity.*
- static const float **NEGATIVE\_INFINITY**  
*Constant for Negative Infinity.*

## 6.364.1 Constructor & Destructor Documentation

### 6.364.1.1 decaf::lang::Float::Float ( float value )

#### Parameters

<i>value</i>	- the primitive type to wrap
--------------	------------------------------



6.364.1.2 decaf::lang::Float::Float ( double *value* )

#### Parameters

<i>value</i>	- the primitive type to wrap
--------------	------------------------------

6.364.1.3 decaf::lang::Float::Float ( const std::string & *value* ) throw ( exceptions::NumberFormatException )

#### Parameters

<i>value</i>	- the string to convert to a primitive type to wrap
--------------	---

6.364.1.4 virtual decaf::lang::Float::~Float ( ) [inline, virtual]

### 6.364.2 Member Function Documentation

6.364.2.1 virtual unsigned char decaf::lang::Float::byteValue ( ) const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p.2787).

6.364.2.2 static int decaf::lang::Float::compare ( float *f1*, float *f2* ) [static]

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: Float( *f1* ).compareTo( Float( *f2* ) )

#### Parameters

<i>f1</i>	- the first double to compare
<i>f2</i>	- the second double to compare

#### Returns

the value 0 if *d1* is numerically equal to *f2*; a value less than 0 if *f1* is numerically less than *f2*; and a value greater than 0 if *f1* is numerically greater than *f2*.

6.364.2.3 virtual int decaf::lang::Float::compareTo ( const float & *f* ) const [virtual]

Compares this **Float** (p. 1865) instance with another.

**Parameters**

<i>f</i>	- the <b>Float</b> (p. 1865) instance to be compared
----------	--

**Returns**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **float** > (p. 1187).

6.364.2.4 `virtual int decaf::lang::Float::compareTo ( const Float & f ) const` [virtual]

Compares this **Float** (p. 1865) instance with another.

**Parameters**

<i>f</i>	- the <b>Float</b> (p. 1865) instance to be compared
----------	--

**Returns**

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Float** > (p. 1187).

6.364.2.5 `virtual double decaf::lang::Float::doubleValue ( ) const` [inline, virtual]

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.364.2.6 `bool decaf::lang::Float::equals ( const Float & f ) const` [inline, virtual]

**Parameters**

<i>f</i>	- the <b>Float</b> (p. 1865) object to compare against.
----------	---

**Returns**

true if the two **Float** (p. 1865) Objects have the same value.

Implements **decaf::lang::Comparable**< **Float** > (p. 1188).

6.364.2.7 `bool decaf::lang::Float::equals ( const float & f ) const [inline, virtual]`

#### Parameters

<i>f</i> - the <b>Float</b> (p. 1865) object to compare against.
--

#### Returns

true if the two **Float** (p. 1865) Objects have the same value.

Implements **decaf::lang::Comparable**< float > (p. 1188).

6.364.2.8 `static int decaf::lang::Float::floatToIntBits ( float value ) [static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7fc00000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1870) method, will produce a floating-point value the same as the argument to **floatToIntBits** (except all NaN values are collapsed to a single "canonical" NaN value).

#### Parameters

<i>value</i> - the float to convert to int bits
---

#### Returns

the int that holds the float's value

6.364.2.9 `static int decaf::lang::Float::floatToRawIntBits ( float value ) [static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the **floatToIntBits** method, **intToRawIntBits** does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1870)

method, will produce a floating-point value the same as the argument to `floatToRawIntBits`.

#### Parameters

<i>value</i>	The float to convert to a raw int.
--------------	------------------------------------

#### Returns

the raw int value of the float

**6.364.2.10** `virtual float decaf::lang::Float::floatValue ( ) const` `[inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

**6.364.2.11** `static float decaf::lang::Float::intBitsToFloat ( int bits )` `[static]`

Returns the float value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Float::floatToRawIntBits** (p. 1870) method.

#### Parameters

<i>bits</i>	- the bits of the float encoded as a float
-------------	--

#### Returns

a new float created from the int bits.

**6.364.2.12** `virtual int decaf::lang::Float::intValue ( ) const` `[inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.364.2.13 `bool decaf::lang::Float::isInfinite ( ) const`

#### Returns

true if the float is equal to positive infinity.

6.364.2.14 `static bool decaf::lang::Float::isInfinite ( float value ) [static]`

#### Parameters

<i>value</i>	- The float to check.
--------------	-----------------------

#### Returns

true if the float is equal to infinity.

6.364.2.15 `bool decaf::lang::Float::isNaN ( ) const`

#### Returns

true if the float is equal to NaN.

6.364.2.16 `static bool decaf::lang::Float::isNaN ( float value ) [static]`

#### Parameters

<i>value</i>	- The float to check.
--------------	-----------------------

#### Returns

true if the float is equal to NaN.

6.364.2.17 `virtual long long decaf::lang::Float::longValue ( ) const [inline,  
virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

**6.364.2.18** `virtual bool decaf::lang::Float::operator< ( const float & f ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< float >** (p. 1188).

**6.364.2.19** `virtual bool decaf::lang::Float::operator< ( const Float & f ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Float >** (p. 1188).

**6.364.2.20** `virtual bool decaf::lang::Float::operator==( const Float & f ) const [inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<i>f</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Float >** (p. 1189).

6.364.2.21 `virtual bool decaf::lang::Float::operator==( const float & f ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<i>f</i> - the value to be compared to this one.
--

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< float > (p. 1189).

6.364.2.22 `static float decaf::lang::Float::parseFloat ( const std::string & value ) throw ( exceptions::NumberFormatException )` `[static]`

Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1865).

#### Parameters

<i>value</i> - the string to parse
------------------------------------

#### Returns

a float parsed from the string

#### Exceptions

<i>NumberFormatException</i>
------------------------------

6.364.2.23 `virtual short decaf::lang::Float::shortValue ( ) const` `[inline, virtual]`

Answers the short value which the receiver represents.

#### Returns

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2788).

6.364.2.24 `static std::string decaf::lang::Float::toHexString ( float value )` `[static]`

Returns a hexadecimal string representation of the float argument.

All characters mentioned below are ASCII characters.

\* If the argument is NaN, the result is the string "NaN". \* Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p.2051) on the exponent value. o If m is a float value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

#### Parameters

<i>value</i>	- The float to convert to a string
--------------	------------------------------------

#### Returns

the Hex formatted float string.

6.364.2.25 `std::string decaf::lang::Float::toString ( ) const`

#### Returns

this **Float** (p. 1865) Object as a **String** (p. 3610) Representation

6.364.2.26 `static std::string decaf::lang::Float::toString ( float value ) [static]`

Returns a string representation of the float argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10<sup>-3</sup> but less than 10<sup>7</sup>, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed



by one or more decimal digits representing the fractional part of  $m$ . o If  $m$  is less than  $10^{-3}$  or greater than or equal to  $10^7$ , then it is represented in so-called "computerized scientific notation." Let  $n$  be the unique integer such that  $10^n \leq m < 10^{n+1}$ ; then let  $a$  be the mathematically exact quotient of  $m$  and  $10^n$  so that  $1 \leq a < 10$ . The magnitude is then represented as the integer part of  $a$ , as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of  $a$ , followed by the letter 'E', followed by a representation of  $n$  as a decimal integer, as produced by the method **Integer.toString(int)** (p. 2052).

#### Parameters

<i>value</i>	- The float to convert to a string
--------------	------------------------------------

#### Returns

the formatted float string.

6.364.2.27 **static Float** decaf::lang::Float::valueOf ( **const std::string & value** ) **throw ( exceptions::NumberFormatException )** [static]

Returns a **Float** (p. 1865) instance that wraps a primitive float which is parsed from the string value passed.

#### Parameters

<i>value</i>	- the string to parse
--------------	-----------------------

#### Returns

a new **Float** (p. 1865) instance wrapping the float parsed from value

#### Exceptions

<i>NumberFormatException</i>	on error.
------------------------------	-----------

6.364.2.28 **static Float** decaf::lang::Float::valueOf ( **float value** ) [static]

Returns a **Float** (p. 1865) instance representing the specified float value.

#### Parameters

<i>value</i>	- float to wrap
--------------	-----------------

#### Returns

new **Float** (p. 1865) instance wrapping the primitive value

### 6.364.3 Field Documentation

6.364.3.1 `const float decaf::lang::Float::MAX_VALUE` `[static]`

The maximum value that the primitive type can hold.

6.364.3.2 `const float decaf::lang::Float::MIN_VALUE` `[static]`

The minimum value that the primitive type can hold.

6.364.3.3 `const float decaf::lang::Float::NaN` `[static]`

Constant for the Not a **Number** (p. 2786) Value.

6.364.3.4 `const float decaf::lang::Float::NEGATIVE_INFINITY` `[static]`

Constant for Negative Infinity.

6.364.3.5 `const float decaf::lang::Float::POSITIVE_INFINITY` `[static]`

Constant for Positive Infinity.

6.364.3.6 `const int decaf::lang::Float::SIZE = 32` `[static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Float.h`

## 6.365 `decaf::internal::nio::FloatArrayBuffer` Class Reference

```
#include <src/main/decaf/internal/nio/FloatArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::FloatArrayBuffer`:

### Public Member Functions

- **`FloatArrayBuffer`** (int size, bool readOnly=false) throw ( `decaf::lang::exceptions::IllegalArgumentException` )

*Creates a **`FloatArrayBuffer`** (p. 1876) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **FloatArrayBuffer** (float \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **FloatArrayBuffer** (p. 1876) object that wraps the given array.*

- **FloatArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int capacity, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*

- **FloatArrayBuffer** (const **FloatArrayBuffer** &other)

*Create a **FloatArrayBuffer** (p. 1876) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**FloatArrayBuffer** ()
- virtual float \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the float array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

#### Returns

*the array that backs this **Buffer** (p. 887).*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this <b>Buffer</b> (p. 887) is read only.</i>
UnsupportedOperationException	<i>if the underlying store has no array.</i>

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

#### Returns

*The offset into the backing array where index zero starts.*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this <b>Buffer</b> (p. 887) is read only.</i>
UnsupportedOperationException	<i>if the underlying store has no array.</i>

- virtual FloatBuffer \* **asReadOnlyBuffer** () const

*Creates a new, read-only float buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will*

not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only float buffer which the caller then owns.

- virtual FloatBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **FloatBuffer** (p. 1887).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only
---	-----------------------------

- virtual FloatBuffer \* **duplicate** ()

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new float **Buffer** (p. 887) which the caller owns.

- virtual float **get** () throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the float at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

- virtual float **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the value at the given index.

**Parameters**

index	The index in the <b>Buffer</b> (p. 887) where the float is to be read
-------	---

**Returns**

the float that is located at the given index

**Exceptions**

IndexOutOfBoundsException	if index is not smaller than the buffer's limit
---------------------------	---

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

- virtual FloatBuffer & **put** (float value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes the given floats into this buffer at the current position, and then increments the position.

**Parameters**

value	The floats value to be written.
-------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

- virtual FloatBuffer & **put** (int index, float value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes the given floats into this buffer at the given index.

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data.
value	The floats to write.

**Returns**

a reference to this buffer.

**Exceptions**

IndexOutOfBoundsException	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
---------------------------	--

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

- virtual `FloatBuffer * slice () const`

Creates a new **FloatBuffer** (p. 1887) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **FloatBuffer** (p. 1887) which the caller owns.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **FloatArrayBuffer** (p. 1876) as Read-Only.

## 6.365.1 Constructor & Destructor Documentation

6.365.1.1 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **FloatArrayBuffer** (p. 1876) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

6.365.1.2 `decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( float * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **FloatArrayBuffer** (p. 1876) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

```
6.365.1.3 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( const
decaf::lang::Pointer< ByteArrayAdapter > & array,
int offset, int capacity, bool readOnly = false ) throw
( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IndexOutOfBoundsException )
```

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **FloatArrayBuffer** (p. 1876) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

```
6.365.1.4 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer ( const FloatArrayBuffer &
other )
```

Create a **FloatArrayBuffer** (p. 1876) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

**Parameters**

<i>other</i>	The <b>FloatArrayBuffer</b> (p. 1876) this one is to mirror.
--------------	--

6.365.1.5 `virtual decaf::internal::nio::FloatArrayBuffer::~FloatArrayBuffer ( )` [virtual]

**6.365.2 Member Function Documentation**

6.365.2.1 `virtual float* decaf::internal::nio::FloatArrayBuffer::array ( )` throw (  
`decaf::lang::exceptions::UnsupportedOperationException,`  
`decaf::nio::ReadOnlyBufferException`) [virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 887).

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1890).

6.365.2.2 `virtual int decaf::internal::nio::FloatArrayBuffer::arrayOffset ( )` throw (  
`decaf::lang::exceptions::UnsupportedOperationException,`  
`decaf::nio::ReadOnlyBufferException`) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**



<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b>UnsupportedOperationException</b>	if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1891).

**6.365.2.3** `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::asReadOnlyBuffer ( )  
const [virtual]`

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p. 1891).

**6.365.2.4** `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact ( ) throw (  
decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **FloatBuffer** (p. 1887).

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only
--	-----------------------------

Implements **decaf::nio::FloatBuffer** (p. 1892).

6.365.2.5 **virtual FloatBuffer\* decaf::internal::nio::FloatArrayBuffer::duplicate ( )**  
[virtual]

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new float **Buffer** (p. 887) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1892).

6.365.2.6 **virtual float decaf::internal::nio::FloatArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException )** [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the float at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return.
--	----------------------------------

Implements **decaf::nio::FloatBuffer** (p. 1894).

6.365.2.7 **virtual float decaf::internal::nio::FloatArrayBuffer::get ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException )** [virtual]

Absolute get method.

Reads the value at the given index.

**Parameters**

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the float is to be read
--------------	---

**Returns**

the float that is located at the given index

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implements **decaf::nio::FloatBuffer** (p. 1893).

```
6.365.2.8  virtual bool decaf::internal::nio::FloatArrayBuffer::hasArray ( ) const  [inline, virtual]
```

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::FloatBuffer** (p. 1894).

```
6.365.2.9  virtual bool decaf::internal::nio::FloatArrayBuffer::isReadOnly ( ) const  [inline, virtual]
```

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 890).

```
6.365.2.10 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put ( int index, float value ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )  [virtual]
```

Writes the given floats into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The floats to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::FloatBuffer** (p. 1895).

6.365.2.11 `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put ( float value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given floats into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The floats value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1895).

6.365.2.12 `virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly ( bool value ) [inline, protected, virtual]`

Sets this **FloatArrayBuffer** (p. 1876) as Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.365.2.13 `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **FloatBuffer** (p. 1887) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **FloatBuffer** (p. 1887) which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1898).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/FloatArrayBuffer.h`

## 6.366 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:

```
#include <src/main/decaf/nio/FloatBuffer.h>
```

Inheritance diagram for `decaf::nio::FloatBuffer`:

### Public Member Functions

- `virtual ~FloatBuffer ()`
- `virtual std::string toString () const`
- `virtual float * array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the float array that backs this buffer (optional operation).*
- `virtual int arrayOffset ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- `virtual FloatBuffer * asReadOnlyBuffer () const =0`  
*Creates a new, read-only float buffer that shares this buffer's content.*
- `virtual FloatBuffer & compact ()=0 throw ( ReadOnlyBufferException )`  
*Compacts this buffer.*

- virtual **FloatBuffer** \* **duplicate** ()=0  
*Creates a new float buffer that shares this buffer's content.*
- virtual float **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual float **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **FloatBuffer** & **get** (std::vector< float > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **FloatBuffer** & **get** (float \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible float array.*
- **FloatBuffer** & **put** (**FloatBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the floats remaining in the given source buffer into this buffer.*
- **FloatBuffer** & **put** (const float \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers floats into this buffer from the given source array.*
- **FloatBuffer** & **put** (std::vector< float > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source floats array into this buffer.*
- virtual **FloatBuffer** & **put** (float value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given floats into this buffer at the current position, and then increments the position.*
- virtual **FloatBuffer** & **put** (int index, float value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given floats into this buffer at the given index.*
- virtual **FloatBuffer** \* **slice** () const =0  
*Creates a new **FloatBuffer** (p. 1887) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **FloatBuffer** &value) const
- virtual bool **equals** (const **FloatBuffer** &value) const
- virtual bool **operator==** (const **FloatBuffer** &value) const
- virtual bool **operator<** (const **FloatBuffer** &value) const

## Static Public Member Functions

- static **FloatBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new Double buffer.*
- static **FloatBuffer** \* **wrap** (float \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **FloatBuffer** (p. 1887).*
- static **FloatBuffer** \* **wrap** (std::vector< float > &buffer)  
*Wraps the passed STL float Vector in a **FloatBuffer** (p. 1887).*

## Protected Member Functions

- **FloatBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **FloatBuffer** (p. 1887) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.366.1 Detailed Description

This class defines four categories of operations upon float buffers:

o Absolute and relative get and put methods that read and write single floats; o Relative bulk get methods that transfer contiguous sequences of floats from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.366.2 Constructor & Destructor Documentation

6.366.2.1 decaf::nio::FloatBuffer::FloatBuffer ( int *capacity* ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **FloatBuffer** (p. 1887) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 887) in floats.
-----------------	---

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if capacity is negative.
---------------------------------------	--------------------------

6.366.2.2 `virtual decaf::nio::FloatBuffer::~~FloatBuffer ( ) [inline, virtual]`

**6.366.3 Member Function Documentation**

6.366.3.1 `static FloatBuffer* decaf::nio::FloatBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

**Parameters**

<i>capacity</i>	The size of the Double buffer in floats.
-----------------	--

**Returns**

the **FloatBuffer** (p. 1887) that was allocated, caller owns.

6.366.3.2 `virtual float* decaf::nio::FloatBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 887).

**Exceptions**

<b><i>ReadOnlyBufferEx- ception</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOpera- tionException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1882).



6.366.3.3 `virtual int decaf::nio::FloatBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1882).

6.366.3.4 `virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer ( ) const [pure virtual]`

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

### Returns

The new, read-only float buffer which the caller then owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1883).

6.366.3.5 `virtual FloatBuffer& decaf::nio::FloatBuffer::compact ( ) throw ( ReadOnlyBufferException ) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **FloatBuffer** (p. 1887).

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only
---	-----------------------------

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1883).

6.366.3.6 `virtual int decaf::nio::FloatBuffer::compareTo ( const FloatBuffer & value ) const`  
[virtual]

6.366.3.7 `virtual FloatBuffer* decaf::nio::FloatBuffer::duplicate ( )` [pure virtual]

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new float **Buffer** (p. 887) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1884).

6.366.3.8 `virtual bool decaf::nio::FloatBuffer::equals ( const FloatBuffer & value ) const`  
[virtual]

6.366.3.9 `FloatBuffer& decaf::nio::FloatBuffer::get ( std::vector< float > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this **Buffer** (p. 887).

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length floats remaining in this buffer
--	--

6.366.3.10 `virtual float decaf::nio::FloatBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` [pure virtual]

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the float is to be read
--------------	---

### Returns

the float that is located at the given index

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit
----------------------------------	---

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1884).

6.366.3.11 `FloatBuffer& decaf::nio::FloatBuffer::get ( float * buffer, int size, int offset, int length ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Relative bulk get method.

This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 892), then no bytes are transferred and a **BufferUnderflowException** (p. 916) is thrown.

Otherwise, this method copies length floats from this buffer into the given array, starting

at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 887).

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length floats remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.366.3.12 `virtual float decaf::nio::FloatBuffer::get ( ) throw ( BufferUnderflowException )`  
[pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the float at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1884).

6.366.3.13 `virtual bool decaf::nio::FloatBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1885).

6.366.3.14 `virtual bool decaf::nio::FloatBuffer::operator< ( const FloatBuffer & value ) const`  
[virtual]

6.366.3.15 `virtual bool decaf::nio::FloatBuffer::operator== ( const FloatBuffer & value ) const`  
[virtual]

6.366.3.16 `virtual FloatBuffer& decaf::nio::FloatBuffer::put ( float value ) throw (`  
**BufferOverflowException**, **ReadOnlyBufferException** ) [pure  
virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The floats value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1886).

6.366.3.17 `virtual FloatBuffer& decaf::nio::FloatBuffer::put ( int index, float value )`  
`throw ( decaf::lang::exceptions::IndexOutOfBoundsException,`  
**ReadOnlyBufferException** ) [pure virtual]

Writes the given floats into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The floats to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written, or index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1885).

6.366.3.18 **FloatBuffer& decaf::nio::FloatBuffer::put ( const float \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

This method transfers floats into this buffer from the given source array.

If there are more floats to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 892), then no floats are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

**Parameters**

<i>buffer</i>	The array from which floats are to be read.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The offset within the array of the first float to be read.
<i>length</i>	The number of floats to be read from the given array.

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

### 6.366.3.19 FloatBuffer& decaf::nio::FloatBuffer::put ( std::vector< float > & *buffer* ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source floats array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>FloatBuffer</b> (p. 1887)
---------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

### 6.366.3.20 FloatBuffer& decaf::nio::FloatBuffer::put ( FloatBuffer & *src* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

This method transfers the floats remaining in the given source buffer into this buffer.

If there are more floats remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 892), then no floats are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `n = src.remaining()` floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take floats from an place in this one.
------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer for the remaining floats in the source buffer
<b>IllegalArgumentException</b>	if the source buffer is this buffer.

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.
--	------------------------------

6.366.3.21 `virtual FloatBuffer* decaf::nio::FloatBuffer::slice ( ) const [pure virtual]`

Creates a new **FloatBuffer** (p. 1887) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **FloatBuffer** (p. 1887) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1887).

6.366.3.22 `virtual std::string decaf::nio::FloatBuffer::toString ( ) const [virtual]`

#### Returns

a `std::string` describing this object

6.366.3.23 `static FloatBuffer* decaf::nio::FloatBuffer::wrap ( float * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new **FloatBuffer** (p. 1887).

The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the array that was passed in.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.



**Returns**

a new **FloatBuffer** (p. 1887) that is backed by buffer, caller owns.

**Exceptions**

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

**6.366.3.24** static **FloatBuffer\*** decaf::nio::FloatBuffer::wrap ( std::vector< float > & *buffer* )  
[static]

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1887).

The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	- The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new **FloatBuffer** (p. 1887) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**FloatBuffer.h**

**6.367 decaf::io::Flushable Class Reference**

A **Flushable** (p. 1899) is a destination of data that can be flushed.

```
#include <src/main/decaf/io/Flushable.h>
```

Inheritance diagram for decaf::io::Flushable:

**Public Member Functions**

- virtual ~**Flushable** ()
- virtual void **flush** ()=0 throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

### 6.367.1 Detailed Description

A **Flushable** (p. 1899) is a destination of data that can be flushed.

The flush method is invoked to write any buffered output to the underlying stream.

#### Since

1.0

### 6.367.2 Constructor & Destructor Documentation

6.367.2.1 `virtual decaf::io::Flushable::~~Flushable ( ) [inline, virtual]`

### 6.367.3 Member Function Documentation

6.367.3.1 `virtual void decaf::io::Flushable::flush ( ) throw ( decaf::io::IOException ) [pure virtual]`

Flushes this stream by writing any buffered output to the underlying stream.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

Implemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3523), **decaf::internal::io::StandardOutput** (p. 3526), **decaf::io::BufferedOutputStream** (p. 901), **decaf::io::FilterOutputStream** (p. 1864), **decaf::io::OutputStream** (p. 2859), and **decaf::io::OutputStreamWriter** (p. 2866).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Flushable.h`

## 6.368 activemq::commands::FlushCommand Class Reference

```
#include <src/main/activemq/commands/FlushCommand.h>
```

Inheritance diagram for `activemq::commands::FlushCommand`:

#### Public Member Functions

- **FlushCommand** ( )
- virtual **~FlushCommand** ( )
- virtual unsigned char **getDataStructureType** ( ) const

*Get the unique identifier that this object and its own Marshaler share.*

- virtual **FlushCommand** \* **cloneDataStructure** () const

*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*

- virtual void **copyDataStructure** (const **DataStructure** \*src)

*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const

*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*

- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_FLUSHCOMMAND** = 15

## 6.368.1 Constructor & Destructor Documentation

6.368.1.1 **activemq::commands::FlushCommand::FlushCommand** ( )

6.368.1.2 **virtual activemq::commands::FlushCommand::~~FlushCommand** ( )  
[virtual]

## 6.368.2 Member Function Documentation

6.368.2.1 **virtual FlushCommand\*** **activemq::commands::FlushCommand::cloneDataStructure**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

## Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.368.2.2 `virtual void activemq::commands::FlushCommand::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 724).

6.368.2.3 `virtual bool activemq::commands::FlushCommand::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 725).

6.368.2.4 `virtual unsigned char activemq::commands::FlushCommand::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements `activemq::commands::DataStructure` (p. 1631).

6.368.2.5 `virtual std::string activemq::commands::FlushCommand::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 729).

## 6.369 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller

---

### Class Reference 1911

6.368.2.6 virtual **Pointer**<**Command**> **activemq::commands::FlushCommand::visit**  
( **activemq::state::CommandVisitor** \* *visitor* ) throw (  
**exceptions::ActiveMQException** ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.368.3 Field Documentation

6.368.3.1 const unsigned char **activemq::commands::FlushCommand::ID\_**  
**FLUSHCOMMAND** = 15 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**FlushCommand.h**

## 6.369 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller

---

### Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1903).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller**:

#### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.369.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1903).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.369.2 Constructor & Destructor Documentation

6.369.2.1 **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.369.2.2 **virtual activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.369.3 Member Function Documentation

6.369.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.369 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller

### Class Reference 1913

6.369.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.369.3.3 virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.369.3.4 virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.369.3.5  virtual int activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.369.3.6  virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).



## 6.370 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller

### Class Reference

1915

6.369.3.7 virtual void activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**FlushCommandMarshaller.h**

## 6.370 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1907).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller:

#### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.370.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1907).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.370.2 Constructor & Destructor Documentation

6.370.2.1 **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.370.2.2 **virtual activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.370.3 Member Function Documentation

6.370.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.370 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller

### Class Reference 1917

6.370.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.370.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.370.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.370.3.5  virtual int activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.370.3.6  virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

## 6.371 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller

### Class Reference

1919

```
6.370.3.7 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**FlushCommandMarshaller.h**

## 6.371 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1911).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**:

#### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.371.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1911).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.371.2 Constructor & Destructor Documentation

6.371.2.1 **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.371.2.2 **virtual activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.371.3 Member Function Documentation

6.371.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.371 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller

### Class Reference 1921

6.371.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::getDataStructureType  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.371.3.3 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseMarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.371.3.4 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.371.3.5  virtual int activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.371.3.6  virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).



## 6.372 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller

### Class Reference 1923

6.371.3.7 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**FlushCommandMarshaller.h**

## 6.372 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1915).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller:

#### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.372.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1915).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.372.2 Constructor & Destructor Documentation

6.372.2.1 **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.372.2.2 **virtual activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.372.3 Member Function Documentation

6.372.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.372 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller

### Class Reference 1925

6.372.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.372.3.3 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.372.3.4 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.372.3.5  virtual int activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.372.3.6  virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

## 6.373 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller

### Class Reference

1927

6.372.3.7 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**FlushCommandMarshaller.h**

## 6.373 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1919).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller:

#### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.373.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1919).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.373.2 Constructor & Destructor Documentation

6.373.2.1 **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.373.2.2 **virtual activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.373.3 Member Function Documentation

6.373.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

### 6.373 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller

#### Class Reference 1929

6.373.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.373.3.3 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.373.3.4 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.373.3.5  virtual int activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.373.3.6  virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).



## 6.374 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller

### Class Reference 1931

6.373.3.7 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**FlushCommandMarshaller.h**

## 6.374 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1923).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller:

#### Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.374.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1923).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.374.2 Constructor & Destructor Documentation

6.374.2.1 **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::FlushCommandMarshaller**  
( ) [inline]

6.374.2.2 **virtual activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::~~FlushCommandMarshaller**  
( ) [inline, virtual]

### 6.374.3 Member Function Documentation

6.374.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::createObject** (  
) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.374 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller

### Class Reference 1933

6.374.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.374.3.3 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.374.3.4 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.374.3.5  virtual int activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.374.3.6  virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.374.3.7 virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**FlushCommandMarshaller.h**

## 6.375 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1927) provides support for formatting LogRecords.

```
#include <src/main/decaf/util/logging/Formatter.h>
```

Inheritance diagram for decaf::util::logging::Formatter:

#### Public Member Functions

- virtual **~Formatter** ()
- virtual std::string **format** (const **LogRecord** &record) const =0  
*Format the given log record and return the formatted string.*
- virtual std::string **formatMessage** (const **LogRecord** &record) const  
*Format the message string from a log record.*
- virtual std::string **getHead** (const **Handler** \*handler DECAF\_UNUSED)  
*Return the header string for a set of formatted records.*
- virtual std::string **getTail** (const **Handler** \*handler DECAF\_UNUSED)  
*Return the tail string for a set of formatted records.*

### 6.375.1 Detailed Description

A **Formatter** (p. 1927) provides support for formatting LogRecords.

Typically each logging **Handler** (p. 1941) will have a **Formatter** (p. 1927) associated with it. The **Formatter** (p. 1927) takes a **LogRecord** (p. 2370) and converts it to a string.

Some formatters (such as the **XMLFormatter** (p. 3988)) need to wrap head and tail strings around a set of formatted records. The `getHeader` and `getTail` methods can be used to obtain these strings.

### 6.375.2 Constructor & Destructor Documentation

6.375.2.1 `virtual decaf::util::logging::Formatter::~Formatter ( ) [inline, virtual]`

### 6.375.3 Member Function Documentation

6.375.3.1 `virtual std::string decaf::util::logging::Formatter::format ( const LogRecord & record ) const [pure virtual]`

Format the given log record and return the formatted string.

#### Parameters

<i>record</i>	The Log Record to Format
---------------	--------------------------

#### Returns

the formatted record.

Implemented in **decaf::util::logging::SimpleFormatter** (p. 3443), and **decaf::util::logging::XMLFormatter** (p. 3989).

6.375.3.2 `virtual std::string decaf::util::logging::Formatter::formatMessage ( const LogRecord & record ) const [virtual]`

Format the message string from a log record.

#### Parameters

<i>record</i>	The Log Record to Format
---------------	--------------------------

#### Returns

the formatted message

6.375.3.3 virtual std::string decaf::util::logging::Formatter::getHead ( const Handler \*handler  
*DECAF\_UNUSED* ) [inline, virtual]

Return the header string for a set of formatted records.

In the default implementation this method should return empty string.

#### Parameters

<i>handler</i>	The target handler, can be NULL.
----------------	----------------------------------

#### Returns

the head string.

6.375.3.4 virtual std::string decaf::util::logging::Formatter::getTail ( const Handler \*handler  
*DECAF\_UNUSED* ) [inline, virtual]

Return the tail string for a set of formatted records.

In the default implementation this method should return empty string

#### Parameters

<i>handler</i>	the target handler, can be null
----------------	---------------------------------

#### Returns

the tail string

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Formatter.h**

## 6.376 decaf::util::concurrent::Future< V > Class Template Reference

A **Future** (p. 1929) represents the result of an asynchronous computation.

```
#include <src/main/decaf/util/concurrent/Future.h>
```

#### Public Member Functions

- virtual ~**Future** ()
- bool **cancel** (bool mayInterruptIfRunning)=0  
*Attempts to cancel execution of this task.*
- bool **isCancelled** () const =0  
*Returns true if this task was canceled before it completed normally.*

- `bool isDone () const =0`

*Returns true if this task completed.*

- `V get ()=0 throw ( CancellationException, InterruptedException, ExecutionException )`

*Waits if necessary for the computation to complete, and then retrieves its result.*

- `V get (long long timeout, TimeUnit unit)=0 throw ( InterruptedException, ExecutionException, TimeoutException )`

*Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.*

### 6.376.1 Detailed Description

```
template<typename V>class decaf::util::concurrent::Future< V >
```

A **Future** (p. 1929) represents the result of an asynchronous computation.

Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method `get` when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the `cancel` method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p. 1929) for the sake of cancellability but not provide a usable result, you can declare types of the form `Future<void*>` and return null as a result of the underlying task.

### 6.376.2 Constructor & Destructor Documentation

```
6.376.2.1 template<typename V > virtual decaf::util::concurrent::Future< V
>::~~Future( ) [inline, virtual]
```

### 6.376.3 Member Function Documentation

```
6.376.3.1 template<typename V > bool decaf::util::concurrent::Future< V >::cancel (
bool mayInterruptIfRunning ) [pure virtual]
```

Attempts to cancel execution of this task.

This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when `cancel` is called, this task should never run. If the task has already started, then the `mayInterruptIfRunning` parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to `isDone()` (p. 1932) will always return true. Subsequent calls to `isCancelled()` (p. 1932) will always return true if this method returned true.



**Parameters**

<i>mayInterruptIfRunning</i>	- true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.
------------------------------	---

**Returns**

false if the task could not be canceled, typically because it has already completed normally; true otherwise

6.376.3.2 `template<typename V> V decaf::util::concurrent::Future< V >::get ( long long timeout, TimeUnit unit ) throw ( InterruptedException, ExecutionException, TimeoutException ) [pure virtual]`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

**Parameters**

<i>timeout</i>	- the maximum time to wait
<i>unit</i>	- the time unit of the timeout argument

**Returns**

the computed result

**Exceptions**

<b>CancellationException</b> (p. 1052)	- if the computation was canceled
<b>ExecutionException</b> (p. 1829)	- if the computation threw an exception
<i>InterruptedException</i>	- if the current thread was interrupted while waiting
<b>TimeoutException</b> (p. 3728)	- if the wait timed out

6.376.3.3 `template<typename V> V decaf::util::concurrent::Future< V >::get ( ) throw ( CancellationException, InterruptedException, ExecutionException ) [pure virtual]`

Waits if necessary for the computation to complete, and then retrieves its result.

**Returns**

the computed result.

**Exceptions**

<b><i>CancellationException</i></b> (p. 1052)	- if the computation was canceled
<b><i>ExecutionException</i></b> (p. 1829)	- if the computation threw an exception
<b><i>InterruptedException</i></b>	- if the current thread was interrupted while waiting

6.376.3.4 `template<typename V > bool decaf::util::concurrent::Future< V >::isCancelled ( ) const [pure virtual]`

Returns true if this task was canceled before it completed normally.

#### Returns

true if this task was canceled before it completed

6.376.3.5 `template<typename V > bool decaf::util::concurrent::Future< V >::isDone ( ) const [pure virtual]`

Returns true if this task completed.

Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

#### Returns

true if this task completed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Future.h`

## 6.377 activemq::transport::correlator::FutureResponse Class Reference

A container that holds a response object.

```
#include <src/main/activemq/transport/correlator/FutureResponse.h>
```

### Public Member Functions

- `FutureResponse ()`
- `virtual ~FutureResponse ()`
- `virtual const Pointer< Response > &getResponse () const`

*Getters for the response property.*

- virtual **Pointer**< **Response** > & **getResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** (unsigned int timeout) const

*Getters for the response property.*

- virtual **Pointer**< **Response** > & **getResponse** (unsigned int timeout)
- virtual void **setResponse** (const **Pointer**< **Response** > &response)

*Setter for the response property.*

### 6.377.1 Detailed Description

A container that holds a response object.

Callers of the `getResponse` method will block until a response has been received unless they call the `getResponse` that takes a timeout.

### 6.377.2 Constructor & Destructor Documentation

6.377.2.1 `activemq::transport::correlator::FutureResponse::FutureResponse ( )`  
[inline]

6.377.2.2 `virtual activemq::transport::correlator::FutureResponse::~~FutureResponse ( )`  
[inline, virtual]

### 6.377.3 Member Function Documentation

6.377.3.1 `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( )`  
const [inline, virtual]

Getters for the response property.

Infinite Wait.

#### Returns

the response object for the request

6.377.3.2 `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( )`  
[inline, virtual]

6.377.3.3 `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( unsigned int timeout )` [inline, virtual]

6.377.3.4 `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ( unsigned int timeout ) const` `[inline, virtual]`

Getters for the response property.

Timed Wait.

#### Parameters

<i>timeout</i>	- time to wait in milliseconds
----------------	--------------------------------

#### Returns

the response object for the request

6.377.3.5 `virtual void activemq::transport::correlator::FutureResponse::setResponse ( const Pointer< Response > & response )` `[inline, virtual]`

Setter for the response property.

#### Parameters

<i>response</i>	the response object for the request.
-----------------	--------------------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/correlator/FutureResponse.h`

## 6.378 decaf::security::GeneralSecurityException Class Reference

```
#include <src/main/decaf/security/GeneralSecurityException.h>
```

Inheritance diagram for `decaf::security::GeneralSecurityException`:

### Public Member Functions

- **GeneralSecurityException** () throw ()  
*Default Constructor.*
- **GeneralSecurityException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **GeneralSecurityException** (const **GeneralSecurityException** &ex) throw ()  
*Copy Constructor.*
- **GeneralSecurityException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **GeneralSecurityException** (const std::exception \***cause**) throw ()

*Constructor.*

- **GeneralSecurityException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **GeneralSecurityException** \* **clone** () const

*Clones this exception.*

- virtual ~**GeneralSecurityException** () throw ()

## 6.378.1 Constructor & Destructor Documentation

6.378.1.1 decaf::security::GeneralSecurityException::GeneralSecurityException ( ) throw ()  
[inline]

Default Constructor.

6.378.1.2 decaf::security::GeneralSecurityException::GeneralSecurityException ( const decaf::lang::Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

### Parameters

ex	An exception that should become this type of Exception
----	--

6.378.1.3 decaf::security::GeneralSecurityException::GeneralSecurityException ( const GeneralSecurityException & ex ) throw () [inline]

Copy Constructor.

### Parameters

ex	An exception that should become this type of Exception
----	--

6.378.1.4 decaf::security::GeneralSecurityException::GeneralSecurityException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.378.1.5 `decaf::security::GeneralSecurityException::GeneralSecurityException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.378.1.6 `decaf::security::GeneralSecurityException::GeneralSecurityException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.378.1.7 `virtual decaf::security::GeneralSecurityException::~~GeneralSecurityException ( ) throw () [inline, virtual]`

**6.378.2 Member Function Documentation**

6.378.2.1 `virtual GeneralSecurityException* decaf::security::GeneralSecurityException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

## 6.379 `decaf::internal::util::GenericResource< T >` Class Template Reference ¶ 1945

### Returns

A deep copy of this exception.

Reimplemented from `decaf::lang::Exception` (p. 1797).

Reimplemented in `decaf::security::cert::CertificateEncodingException` (p. 1061), `decaf::security::cert::CertificateException` (p. 1062), `decaf::security::cert::CertificateExpiredException` (p. 1064), `decaf::security::cert::CertificateNotYetValidException` (p. 1066), `decaf::security::cert::CertificateParsingException` (p. 1068), `decaf::security::InvalidKeyException` (p. 2096), `decaf::security::KeyException` (p. 2257), `decaf::security::KeyManagementException` (p. 2260), `decaf::security::NoSuchAlgorithmException` (p. 2778), `decaf::security::NoSuchProviderException` (p. 2783), and `decaf::security::SignatureException` (p. 3442).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/GeneralSecurityException.h`

## 6.379 `decaf::internal::util::GenericResource< T >` Class Template Reference

A Generic **Resource** (p. 3223) wraps some type and will delete it when the **Resource** (p. 3223) itself is deleted.

```
#include <src/main/decaf/internal/util/GenericResource.h>
```

Inheritance diagram for `decaf::internal::util::GenericResource< T >`:

### Public Member Functions

- **GenericResource** (T \*value)
- virtual **~GenericResource** ()
- T \* **getManaged** () const
- void **setManaged** (T \*value)

### 6.379.1 Detailed Description

```
template<typename T>class decaf::internal::util::GenericResource< T >
```

A Generic **Resource** (p. 3223) wraps some type and will delete it when the **Resource** (p. 3223) itself is deleted.

### Since

1.0

### 6.379.2 Constructor & Destructor Documentation

6.379.2.1 `template<typename T> decaf::internal::util::GenericResource< T  
>::GenericResource ( T * value ) [inline, explicit]`

6.379.2.2 `template<typename T> virtual decaf::internal::util::GenericResource< T  
>::~~GenericResource ( ) [inline, virtual]`

### 6.379.3 Member Function Documentation

6.379.3.1 `template<typename T> T* decaf::internal::util::GenericResource< T  
>::getManaged ( ) const [inline]`

6.379.3.2 `template<typename T> void decaf::internal::util::GenericResource< T  
>::setManaged ( T * value ) [inline]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/GenericResource.h`

## 6.380 gz\_header\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

### Data Fields

- `int text`
- `uLong time`
- `int xflags`
- `int os`
- `Bytef * extra`
- `uInt extra_len`
- `uInt extra_max`
- `Bytef * name`
- `uInt name_max`
- `Bytef * comment`
- `uInt comm_max`
- `int hcrc`
- `int done`

### 6.380.1 Field Documentation

6.380.1.1 `uInt gz_header_s::comm_max`



6.380.1.2 `Bytef* gz_header_s::comment`

6.380.1.3 `int gz_header_s::done`

6.380.1.4 `Bytef* gz_header_s::extra`

6.380.1.5 `ulnt gz_header_s::extra_len`

6.380.1.6 `ulnt gz_header_s::extra_max`

6.380.1.7 `int gz_header_s::hcrc`

6.380.1.8 `Bytef* gz_header_s::name`

6.380.1.9 `ulnt gz_header_s::name_max`

6.380.1.10 `int gz_header_s::os`

6.380.1.11 `int gz_header_s::text`

6.380.1.12 `uLong gz_header_s::time`

6.380.1.13 `int gz_header_s::xflags`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/zlib.h`

## 6.381 gz\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/gzguts.h>
```

### Data Fields

- `int mode`
- `int fd`
- `char * path`
- `z_off64_t pos`
- `unsigned size`
- `unsigned want`
- `unsigned char * in`
- `unsigned char * out`
- `unsigned char * next`
- `unsigned have`
- `int eof`
- `z_off64_t start`

- `z_off64_t raw`
- `int how`
- `int direct`
- `int level`
- `int strategy`
- `z_off64_t skip`
- `int seek`
- `int err`
- `char * msg`
- `z_stream strm`

### 6.381.1 Field Documentation

6.381.1.1 `int gz_state::direct`

6.381.1.2 `int gz_state::eof`

6.381.1.3 `int gz_state::err`

6.381.1.4 `int gz_state::fd`

6.381.1.5 `unsigned gz_state::have`

6.381.1.6 `int gz_state::how`

6.381.1.7 `unsigned char* gz_state::in`

6.381.1.8 `int gz_state::level`

6.381.1.9 `int gz_state::mode`

6.381.1.10 `char* gz_state::msg`

6.381.1.11 `unsigned char* gz_state::next`

6.381.1.12 `unsigned char* gz_state::out`

6.381.1.13 `char* gz_state::path`

6.381.1.14 `z_off64_t gz_state::pos`

6.381.1.15 `z_off64_t gz_state::raw`

6.381.1.16 `int gz_state::seek`

6.381.1.17 `unsigned gz_state::size`

6.381.1.18 `z_off64_t gz_state::skip`

6.381.1.19 `z_off64_t gz_state::start`

6.381.1.20 `int gz_state::strategy`

6.381.1.21 `z_stream gz_state::strm`

6.381.1.22 `unsigned gz_state::want`

The documentation for this struct was generated from the following file:

- `src/main/decaf/internal/util/zip/gzguts.h`

## 6.382 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1941) object takes log messages from a **Logger** (p. 2345) and exports them.

```
#include <src/main/decaf/util/logging/Handler.h>
```

Inheritance diagram for `decaf::util::logging::Handler`:

### Public Member Functions

- **Handler** ()
- virtual `~Handler` ()
- virtual void **flush** ()=0  
*Flush the Handler's output, clears any buffers.*
- virtual void **publish** (const **LogRecord** &record)=0  
*Publish the Log Record to this **Handler** (p. 1941).*
- virtual bool **isLoggable** (const **LogRecord** &record) const  
*Check if this **Handler** (p. 1941) would actually log a given **LogRecord** (p. 2370).*
- virtual void **setFilter** (**Filter** \*filter)  
*Sets the **Filter** (p. 1853) that this **Handler** (p. 1941) uses to filter Log Records.*
- virtual **Filter** \* **getFilter** ()  
*Gets the **Filter** (p. 1853) that this **Handler** (p. 1941) uses to filter Log Records.*
- virtual void **setLevel** (const **Level** &value)  
***Set** (p. 3379) the log level specifying which message levels will be logged by this **Handler** (p. 1941).*
- virtual **Level** **getLevel** ()  
*Get the log level specifying which message levels will be logged by this **Handler** (p. 1941).*

- virtual void **setFormatter** (**Formatter** \*formatter)  
*Sets the **Formatter** (p. 1927) used by this **Handler** (p. 1941).*
- virtual **Formatter** \* **getFormatter** ()  
*Gets the **Formatter** (p. 1927) used by this **Handler** (p. 1941).*
- virtual void **setErrorManager** (**ErrorManager** \*errorManager)  
*Sets the **Formatter** (p. 1927) used by this **Handler** (p. 1941).*
- virtual **ErrorManager** \* **getErrorManager** ()  
*Gets the **ErrorManager** (p. 1792) used by this **Handler** (p. 1941).*

### Protected Member Functions

- void **reportError** (const std::string &message, **decaf::lang::Exception** \*ex, int code)  
*Protected convenience method to report an error to this Handler's **ErrorManager** (p. 1792).*

#### 6.382.1 Detailed Description

A **Handler** (p. 1941) object takes log messages from a **Logger** (p. 2345) and exports them.

It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log, or whatever.

A **Handler** (p. 1941) can be disabled by doing a **setLevel**(**Level.OFF** (p. 2295)) and can be re-enabled by doing a **setLevel** with an appropriate level.

**Handler** (p. 1941) classes typically use **LogManager** (p. 2363) properties to set default values for the Handler's **Filter** (p. 1853), **Formatter** (p. 1927), and **Level** (p. 2290). See the specific documentation for each concrete **Handler** (p. 1941) class.

#### 6.382.2 Constructor & Destructor Documentation

6.382.2.1 **decaf::util::logging::Handler::Handler** ( )

6.382.2.2 virtual **decaf::util::logging::Handler::~~Handler** ( ) [virtual]

#### 6.382.3 Member Function Documentation

6.382.3.1 virtual void **decaf::util::logging::Handler::flush** ( ) [pure virtual]

Flush the Handler's output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p. 3593).

6.382.3.2 `virtual ErrorManager* decaf::util::logging::Handler::getErrorMessageManager ( )`  
`[inline, virtual]`

Gets the **ErrorManager** (p. 1792) used by this **Handler** (p. 1941).

#### Returns

**ErrorManager** (p. 1792) derived pointer or NULL.

6.382.3.3 `virtual Filter* decaf::util::logging::Handler::getFilter ( )` `[inline, virtual]`

Gets the **Filter** (p. 1853) that this **Handler** (p. 1941) uses to filter Log Records.

#### Returns

**Filter** (p. 1853) derived instance

6.382.3.4 `virtual Formatter* decaf::util::logging::Handler::getFormatter ( )` `[inline, virtual]`

Gets the **Formatter** (p. 1927) used by this **Handler** (p. 1941).

#### Returns

**Filter** (p. 1853) derived instance

6.382.3.5 `virtual Level decaf::util::logging::Handler::getLevel ( )` `[inline, virtual]`

Get the log level specifying which message levels will be logged by this **Handler** (p. 1941).

#### Returns

**Level** (p. 2290) enumeration value

6.382.3.6 `virtual bool decaf::util::logging::Handler::isLoggable ( const LogRecord & record )`  
`const [virtual]`

Check if this **Handler** (p. 1941) would actually log a given **LogRecord** (p. 2370).

This method checks if the **LogRecord** (p. 2370) has an appropriate **Level** (p. 2290) and whether it satisfies any **Filter** (p. 1853). It also may make other **Handler** (p. 1941) specific checks that might prevent a handler from logging the **LogRecord** (p. 2370).

#### Parameters

<i>record</i>	<b>LogRecord</b> (p. 2370) to check
---------------	-------------------------------------

Reimplemented in **decaf::util::logging::StreamHandler** (p. 3594).

6.382.3.7 `virtual void decaf::util::logging::Handler::publish ( const LogRecord & record )`  
`[pure virtual]`

Publish the Log Record to this **Handler** (p. 1941).

#### Parameters

<i>record</i>	The Log Record to Publish
---------------	---------------------------

Implemented in **decaf::util::logging::ConsoleHandler** (p. 1368), and **decaf::util::logging::StreamHandler** (p. 3594).

6.382.3.8 `void decaf::util::logging::Handler::reportError ( const std::string & message,`  
`decaf::lang::Exception * ex, int code )` `[protected]`

Protected convenience method to report an error to this Handler's **ErrorManager** (p. 1792).

#### Parameters

<i>message</i>	- a descriptive string (may be empty)
<i>ex</i>	- an exception (may be NULL)
<i>code</i>	- an error code defined in <b>ErrorManager</b> (p. 1792)

6.382.3.9 `virtual void decaf::util::logging::Handler::setErrorHandler ( ErrorManager *`  
`errorManager )` `[virtual]`

Sets the **Formatter** (p. 1927) used by this **Handler** (p. 1941).

The **ErrorManager**'s "error" method will be invoked if any errors occur while using this **Handler** (p. 1941).

#### Parameters

<i>errorManager</i>	<b>ErrorManager</b> (p. 1792) derived instance
---------------------	--

6.382.3.10 `virtual void decaf::util::logging::Handler::setFilter ( Filter * filter )` `[inline,`  
`virtual]`

Sets the **Filter** (p. 1853) that this **Handler** (p. 1941) uses to filter Log Records.

For each call of publish the **Handler** (p. 1941) will call this **Filter** (p. 1853) (if it is non-null)

to check if the **LogRecord** (p. 2370) should be published or discarded.

#### Parameters

<i>filter</i>	<b>Filter</b> (p. 1853) derived instance
---------------	--

6.382.3.11 virtual void decaf::util::logging::Handler::setFormatter ( **Formatter** \* *formatter* )  
[virtual]

Sets the **Formatter** (p. 1927) used by this **Handler** (p. 1941).

Some Handlers may not use Formatters, in which case the **Formatter** (p. 1927) will be remembered, but not used.

#### Parameters

<i>formatter</i>	<b>Filter</b> (p. 1853) derived instance
------------------	--

6.382.3.12 virtual void decaf::util::logging::Handler::setLevel ( const **Level** & *value* )  
[inline, virtual]

**Set** (p. 3379) the log level specifying which message levels will be logged by this **Handler** (p. 1941).

The intention is to allow developers to turn on voluminous logging, but to limit the messages that are sent to certain Handlers.

#### Parameters

<i>value</i>	<b>Level</b> (p. 2290) enumeration value
--------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Handler.h**

## 6.383 decaf::internal::util::HexStringParser Class Reference

```
#include <src/main/decaf/internal/util/HexStringParser.h>
```

### Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)  
*Create a new HexParser.*
- virtual ~**HexStringParser** ()
- long long **parse** (const std::string &hexString)

*Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.*

### Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)
- static float **parseFloat** (const std::string &hexString)

### 6.383.1 Constructor & Destructor Documentation

6.383.1.1 `decaf::internal::util::HexStringParser::HexStringParser ( int exponentWidth, int mantissaWidth )`

Create a new HexParser.

#### Parameters

<i>exponentWidth</i>	- Width of the exponent for the type to parse
<i>mantissaWidth</i>	- Width of the mantissa for the type to parse

6.383.1.2 `virtual decaf::internal::util::HexStringParser::~~HexStringParser ( ) [inline, virtual]`

### 6.383.2 Member Function Documentation

6.383.2.1 `long long decaf::internal::util::HexStringParser::parse ( const std::string & hexString )`

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

#### Parameters

<i>hexString</i>	- string to parse
------------------	-------------------

#### Returns

the bits parsed from the string

6.383.2.2 `static double decaf::internal::util::HexStringParser::parseDouble ( const std::string & hexString ) [static]`



6.383.2.3 static float decaf::internal::util::HexStringParser::parseFloat ( const std::string & hexString ) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/HexStringParser.h

## 6.384 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p. 1947) class maps hexadecimal strings to the value of an index into the table, i.e.

```
#include <src/main/activemq/wireformat/openwire/utils/HexTable.h>
```

### Public Member Functions

- **HexTable** ()
- virtual **~HexTable** ()
- virtual const std::string & **operator[]** (std::size\_t index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.*
- virtual const std::string & **operator[]** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual std::size\_t **size** () const  
*Returns the max size of this Table.*

### 6.384.1 Detailed Description

The **HexTable** (p. 1947) class maps hexadecimal strings to the value of an index into the table, i.e.

the class will return "FF" for the index 255 in the table.

### 6.384.2 Constructor & Destructor Documentation

6.384.2.1 activemq::wireformat::openwire::utils::HexTable::HexTable ( )

6.384.2.2 virtual activemq::wireformat::openwire::utils::HexTable::~~HexTable ( )  
[inline, virtual]

### 6.384.3 Member Function Documentation

```
6.384.3.1 virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (
    std::size_t index ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException
    ) [virtual]
```

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

#### Parameters

<i>index</i>	The index of the value in the table to fetch.
--------------	---

#### Returns

string containing the hex value if the index

#### Exceptions

<i>IndexOutOfBoundsException</i>	
----------------------------------	--

```
6.384.3.2 virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[]
    ( std::size_t index ) const throw ( de-
    ccaf::lang::exceptions::IndexOutOfBoundsException )
    [virtual]
```

```
6.384.3.3 virtual std::size_t activemq::wireformat::openwire::utils::HexTable::size ( ) const
    [inline, virtual]
```

Returns the max size of this Table.

#### Returns

an integer size value

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**HexTable.h**

## 6.385 decaf::net::HttpRetryException Class Reference

```
#include <src/main/decaf/net/HttpRetryException.h>
```

Inheritance diagram for decaf::net::HttpRetryException:

#### Public Member Functions

- **HttpRetryException** () throw ()

*Default Constructor.*

- **HttpRetryException** (const Exception &ex) throw ()

*Conversion Constructor from some other Exception.*

- **HttpRetryException** (const **HttpRetryException** &ex) throw ()

*Copy Constructor.*

- **HttpRetryException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **HttpRetryException** (const std::exception \*cause) throw ()

*Constructor.*

- **HttpRetryException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **HttpRetryException** \* clone () const

*Clones this exception.*

- virtual ~**HttpRetryException** () throw ()

## 6.385.1 Constructor & Destructor Documentation

### 6.385.1.1 decaf::net::HttpRetryException::HttpRetryException ( ) throw () [inline]

Default Constructor.

### 6.385.1.2 decaf::net::HttpRetryException::HttpRetryException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

### 6.385.1.3 decaf::net::HttpRetryException::HttpRetryException ( const HttpRetryException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.385.1.4 `decaf::net::HttpRetryException::HttpRetryException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.385.1.5 `decaf::net::HttpRetryException::HttpRetryException ( const std::exception * cause )`  
`throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.385.1.6 `decaf::net::HttpRetryException::HttpRetryException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()`  
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.385.1.7 `virtual decaf::net::HttpRetryException::~HttpRetryException ( ) throw ()`  
`[inline, virtual]`

## 6.385.2 Member Function Documentation

6.385.2.1 virtual **HttpRetryException\*** decaf::net::HttpRetryException::clone ( ) const  
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**HttpRetryException.h**

## 6.386 activemq::util::IdGenerator Class Reference

```
#include <src/main/activemq/util/IdGenerator.h>
```

### Data Structures

- class **StaticData**

### Public Member Functions

- **IdGenerator** ()
- **IdGenerator** (const std::string &prefix)
- virtual  $\sim$ **IdGenerator** ()
- std::string **generateId** () const

### Static Public Member Functions

- static std::string **getHostname** ()  
*Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.*
- static std::string **getSeedFromId** (const std::string &id)  
*Gets the seed value from a Generated Id, the count portion is removed.*
- static long long **getSequenceFromId** (const std::string &id)  
*Gets the count value from a Generated Id, the seed portion is removed.*
- static int **compare** (const std::string &id1, const std::string &id2)  
*Compares two generated id values.*

## 6.386.1 Constructor & Destructor Documentation

6.386.1.1 `activemq::util::IdGenerator::IdGenerator ( )`

6.386.1.2 `activemq::util::IdGenerator::IdGenerator ( const std::string & prefix )`

6.386.1.3 `virtual activemq::util::IdGenerator::~~IdGenerator ( )` `[virtual]`

## 6.386.2 Member Function Documentation

6.386.2.1 `static int activemq::util::IdGenerator::compare ( const std::string & id1, const std::string & id2 )` `[static]`

Compares two generated id values.

### Parameters

<i>id1</i>	The first id to compare, or left hand side.
<i>id2</i>	The second id to compare, or right hand side.

### Returns

zero if ids are equal or positive if  $id1 > id2$ ...

6.386.2.2 `std::string activemq::util::IdGenerator::generateId ( )` `const`

### Returns

a newly generated unique id.

6.386.2.3 `static std::string activemq::util::IdGenerator::getHostname ( )` `[static]`

Since the initialization of this object results in the retrieval of the machine's host name we can quickly return it here.

### Returns

the previously retrieved host name.

6.386.2.4 `static std::string activemq::util::IdGenerator::getSeedFromId ( const std::string & id )` `[static]`

Gets the seed value from a Generated Id, the count portion is removed.

### Returns

the seed portion of the Id, minus the count value.

## 6.387 decaf::lang::exceptions::IllegalArgumentException Class Reference 1961

6.386.2.5 static long long activemq::util::IdGenerator::getSequenceFromId ( const std::string & id ) [static]

Gets the count value from a Generated Id, the seed portion is removed.

### Returns

the sequence count portion of the id, minus the seed value.

The documentation for this class was generated from the following file:

- src/main/activemq/util/IdGenerator.h

## 6.387 decaf::lang::exceptions::IllegalArgumentException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalArgumentException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalArgumentException:

### Public Member Functions

- **IllegalArgumentException** () throw ()  
*Default Constructor.*
- **IllegalArgumentException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()  
*Copy Constructor.*
- **IllegalArgumentException** (const std::exception \*cause) throw ()  
*Constructor.*
- **IllegalArgumentException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalArgumentException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **IllegalArgumentException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalArgumentException** () throw ()

### 6.387.1 Constructor & Destructor Documentation

6.387.1.1 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( )  
throw () [inline]`

Default Constructor.

6.387.1.2 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
Exception & ex ) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.387.1.3 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
IllegalArgumentException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.387.1.4 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.387.1.5 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const  
char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.



## 6.387 decaf::lang::exceptions::IllegalArgumentException Class Reference 1963

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.387.1.6 `decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.387.1.7 `virtual decaf::lang::exceptions::IllegalArgumentException::~IllegalArgumentException ( ) throw () [inline, virtual]`

## 6.387.2 Member Function Documentation

6.387.2.1 `virtual IllegalArgumentException* decaf::lang::exceptions::IllegalArgumentException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalArgumentException.h`

## 6.388 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalMonitorStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalMonitorStateException:

### Public Member Functions

- **IllegalMonitorStateException** () throw ()  
*Default Constructor.*
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalMonitorStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalMonitorStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalMonitorStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalMonitorStateException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalMonitorStateException** () throw ()

### 6.388.1 Constructor & Destructor Documentation

6.388.1.1 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( ) throw () [inline]

Default Constructor.

6.388.1.2 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const **Exception** & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

ex	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
----	--

## 6.388 decaf::lang::exceptions::IllegalMonitorStateException Class Reference ¶ 965

6.388.1.3 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const IllegalMonitorStateException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.388.1.4 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.388.1.5 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.388.1.6 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

**Parameters**

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.388.1.7 `virtual decaf::lang::exceptions::IllegalMonitorStateException::~~IllegalMonitorStateException ( ) throw () [inline, virtual]`

**6.388.2 Member Function Documentation**

6.388.2.1 `virtual IllegalMonitorStateException* decaf::lang::exceptions::IllegalMonitorStateException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalMonitorStateException.h`

**6.389 cms::IllegalStateException Class Reference**

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

```
#include <src/main/cms/IllegalStateException.h>
```

Inheritance diagram for cms::IllegalStateException:

**Public Member Functions**

- **IllegalStateException** ( ) throw ( )
- **IllegalStateException** (const **IllegalStateException** &ex) throw ( )
- **IllegalStateException** (const std::string &message, const std::exception \*cause) throw ( )
- **IllegalStateException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ( )
- virtual **~IllegalStateException** ( ) throw ( )

### 6.389.1 Detailed Description

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

For example, this exception must be thrown if **Session.commit** (p. 3309) is called on a non-transacted session.

#### Since

1.3

### 6.389.2 Constructor & Destructor Documentation

- 6.389.2.1 `cms::IllegalStateException::IllegalStateException ( ) throw ()`
- 6.389.2.2 `cms::IllegalStateException::IllegalStateException ( const IllegalStateException & ex ) throw ()`
- 6.389.2.3 `cms::IllegalStateException::IllegalStateException ( const std::string & message, const std::exception * cause ) throw ()`
- 6.389.2.4 `cms::IllegalStateException::IllegalStateException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`
- 6.389.2.5 `virtual cms::IllegalStateException::~~IllegalStateException ( ) throw ()`  
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/IllegalStateException.h`

## 6.390 decaf::lang::exceptions::IllegalStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::IllegalStateException:

### Public Member Functions

- **IllegalStateException** () throw ()  
*Default Constructor.*
- **IllegalStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*

- **IllegalStateException** (const **IllegalStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalStateException** \* clone () const  
*Clones this exception.*
- virtual ~**IllegalStateException** () throw ()

### 6.390.1 Constructor & Destructor Documentation

6.390.1.1 `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( ) throw ()` `[inline]`

Default Constructor.

6.390.1.2 `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

<code>ex</code>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------------	--

6.390.1.3 `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const IllegalStateException & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<code>ex</code>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------------	--

6.390.1.4 `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.390.1.5 `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.390.1.6 `decaf::lang::exceptions::IllegalStateException::IllegalStateException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.390.1.7 `virtual decaf::lang::exceptions::IllegalStateException::~~IllegalStateException ( ) throw () [inline, virtual]`

### 6.390.2 Member Function Documentation

6.390.2.1 `virtual IllegalStateException* decaf::lang::exceptions::IllegalStateException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 2099).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalStateException.h`

## 6.391 decaf::lang::exceptions::IllegalThreadStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalThreadStateException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalThreadStateException`:

### Public Member Functions

- **IllegalThreadStateException** () throw ()  
*Default Constructor.*
- **IllegalThreadStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **IllegalThreadStateException** (const **IllegalThreadStateException** &ex) throw ()  
*Copy Constructor.*
- **IllegalThreadStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalThreadStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IllegalThreadStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IllegalThreadStateException** \* **clone** () const  
*Clones this exception.*
- virtual ~**IllegalThreadStateException** () throw ()



## 6.391 decaf::lang::exceptions::IllegalThreadStateException Class Reference 1971

### 6.391.1 Constructor & Destructor Documentation

6.391.1.1 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( )  
throw () [inline]

Default Constructor.

6.391.1.2 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.391.1.3 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const IllegalThreadStateException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.391.1.4 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.391.1.5 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.391.1.6 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.391.1.7 `virtual decaf::lang::exceptions::IllegalThreadStateException::~~IllegalThreadStateException ( ) throw () [inline, virtual]`

## 6.391.2 Member Function Documentation

6.391.2.1 `virtual IllegalThreadStateException* decaf::lang::exceptions::IllegalThreadStateException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalThreadStateException.h`

## 6.392 activemq::transport::inactivity::InactivityMonitor Class Reference

```
#include <src/main/activemq/transport/inactivity/InactivityMonitor.h>
```

Inheritance diagram for activemq::transport::inactivity::InactivityMonitor:

### Public Member Functions

- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **decaf::util::Properties** &properties, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- virtual ~**InactivityMonitor** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- bool **isKeepAliveResponseRequired** () const
- void **setKeepAliveResponseRequired** (bool value)
- long long **getReadCheckTime** () const
- void **setReadCheckTime** (long long value)
- long long **getWriteCheckTime** () const
- void **setWriteCheckTime** (long long value)
- long long **getInitialDelayTime** () const
- void **setInitialDelayTime** (long long value) const

### Friends

- class **ReadChecker**
- class **AsyncSignalReadErrorTask**
- class **WriteChecker**
- class **AsyncWriteTask**

### 6.392.1 Constructor & Destructor Documentation

6.392.1.1 `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor ( const Pointer< Transport > & next, const Pointer< wireformat::WireFormat > & wireFormat )`

6.392.1.2 `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor ( const Pointer< Transport > & next, const decaf::util::Properties & properties, const Pointer< wireformat::WireFormat > & wireFormat )`

6.392.1.3 `virtual activemq::transport::inactivity::InactivityMonitor::~~InactivityMonitor ( )`  
[virtual]

### 6.392.2 Member Function Documentation

6.392.2.1 `virtual void activemq::transport::inactivity::InactivityMonitor::close ( ) throw ( decaf::io::IOException )` [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions

<i>IOException</i>	if an error occurs while closing the <b>Transport</b> (p. 3819).
--------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 3829).

6.392.2.2 `long long activemq::transport::inactivity::InactivityMonitor::getInitialDelayTime ( )`  
const

6.392.2.3 `long long activemq::transport::inactivity::InactivityMonitor::getReadCheckTime ( )`  
const

6.392.2.4 `long long activemq::transport::inactivity::InactivityMonitor::getWriteCheckTime ( )`  
const

6.392.2.5 `bool activemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired ( )` const

6.392.2.6 `virtual void activemq::transport::inactivity::InactivityMonitor::onCommand ( const Pointer< Command > & command )` [virtual]

Event handler for the receipt of a command.

#### Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

6.392.2.7 `virtual void activemq::transport::inactivity::InactivityMonitor::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
[virtual]

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

6.392.2.8 `virtual void activemq::transport::inactivity::InactivityMonitor::onException ( const decaf::lang::Exception & ex )` [virtual]

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception to handle.
-----------	--------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

6.392.2.9 `void activemq::transport::inactivity::InactivityMonitor::setInitialDelayTime ( long long value ) const`

6.392.2.10 `void activemq::transport::inactivity::InactivityMonitor::setKeepAliveResponseRequired ( bool value )`

6.392.2.11 `void activemq::transport::inactivity::InactivityMonitor::setReadCheckTime ( long long value )`

6.392.2.12 `void activemq::transport::inactivity::InactivityMonitor::setWriteCheckTime ( long long value )`

### 6.392.3 Friends And Related Function Documentation

6.392.3.1 friend class AsyncSignalReadErrorTask [friend]

6.392.3.2 friend class AsyncWriteTask [friend]

6.392.3.3 friend class ReadChecker [friend]

6.392.3.4 friend class WriteChecker [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/inactivity/InactivityMonitor.h

## 6.393 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

```
#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Inheritance diagram for decaf::lang::exceptions::IndexOutOfBoundsException:

### Public Member Functions

- **IndexOutOfBoundsException** () throw ()  
*Default Constructor.*
- **IndexOutOfBoundsException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **IndexOutOfBoundsException** (const **IndexOutOfBoundsException** &ex) throw ()  
*Copy Constructor.*
- **IndexOutOfBoundsException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IndexOutOfBoundsException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IndexOutOfBoundsException** (const std::exception \*cause) throw ()  
*Constructor.*
- virtual **IndexOutOfBoundsException** \* **clone** () const  
*Clones this exception.*
- virtual ~**IndexOutOfBoundsException** () throw ()

## 6.393 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference 4977

### 6.393.1 Constructor & Destructor Documentation

6.393.1.1 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( ) throw () [inline]`

Default Constructor.

6.393.1.2 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.393.1.3 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const IndexOutOfBoundsException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.393.1.4 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.393.1.5 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.393.1.6 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.393.1.7 `virtual decaf::lang::exceptions::IndexOutOfBoundsException::~~IndexOutOfBoundsException ( ) throw () [inline, virtual]`

### 6.393.2 Member Function Documentation

6.393.2.1 `virtual IndexOutOfBoundsException* decaf::lang::exceptions::IndexOutOfBoundsException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h`

## 6.394 decaf::net::Inet4Address Class Reference

```
#include <src/main/decaf/net/Inet4Address.h>
```



Inheritance diagram for decaf::net::Inet4Address:

## Public Member Functions

- virtual **~Inet4Address** ()
- virtual bool **isAnyLocalAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid wildcard address.*
- virtual bool **isLoopbackAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid loopback address.*
- virtual bool **isMulticastAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid Multicast address.*
- virtual bool **isLinkLocalAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid link local address.*
- virtual bool **isSiteLocalAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid site local address.*
- virtual bool **isMCGlobal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Global scope.*
- virtual bool **isMCNodeLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Node Local scope.*
- virtual bool **isMCLinkLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Link Local scope.*
- virtual bool **isMCSiteLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Site Local scope.*
- virtual bool **isMCOrgLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Organization scope.*

## Protected Member Functions

- **Inet4Address** ()
- **Inet4Address** (const unsigned char \*ipAddress, int numBytes)
- **Inet4Address** (const std::string &hostname, const unsigned char \*ipAddress, int numBytes)

## Friends

- class **InetAddress**

### 6.394.1 Constructor & Destructor Documentation

- 6.394.1.1 `decaf::net::Inet4Address::Inet4Address ( )` [protected]
- 6.394.1.2 `decaf::net::Inet4Address::Inet4Address ( const unsigned char * ipAddress, int numBytes )` [protected]
- 6.394.1.3 `decaf::net::Inet4Address::Inet4Address ( const std::string & hostname, const unsigned char * ipAddress, int numBytes )` [protected]
- 6.394.1.4 `virtual decaf::net::Inet4Address::~~Inet4Address ( )` [virtual]

### 6.394.2 Member Function Documentation

- 6.394.2.1 `virtual bool decaf::net::Inet4Address::isAnyLocalAddress ( ) const` [virtual]

Check if this **InetAddress** (p. 1974) is a valid wildcard address.

#### Returns

true if the address is a wildcard address.

Reimplemented from **decaf::net::InetAddress** (p. 1979).

- 6.394.2.2 `virtual bool decaf::net::Inet4Address::isLinkLocalAddress ( ) const` [virtual]

Check if this **InetAddress** (p. 1974) is a valid link local address.

#### Returns

true if the address is a link local address.

Reimplemented from **decaf::net::InetAddress** (p. 1979).

- 6.394.2.3 `virtual bool decaf::net::Inet4Address::isLoopbackAddress ( ) const` [virtual]

Check if this **InetAddress** (p. 1974) is a valid loopback address.

#### Returns

true if the address is a loopback address.

Reimplemented from **decaf::net::InetAddress** (p. 1979).

- 6.394.2.4 `virtual bool decaf::net::Inet4Address::isMCGlobal ( ) const` [virtual]

Check if this **InetAddress** (p. 1974) is Multicast and has Global scope.

**Returns**

true if the address is Multicast and has Global scope.

Reimplemented from **decaf::net::InetAddress** (p. 1980).

6.394.2.5 virtual bool decaf::net::Inet4Address::isMCLinkLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1974) is Multicast and has Link Local scope.

**Returns**

true if the address is Multicast and has Link Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1980).

6.394.2.6 virtual bool decaf::net::Inet4Address::isMCNodeLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1974) is Multicast and has Node Local scope.

**Returns**

true if the address is Multicast and has Node Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1980).

6.394.2.7 virtual bool decaf::net::Inet4Address::isMCOrgLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1974) is Multicast and has Organization scope.

**Returns**

true if the address is Multicast and has Organization scope.

Reimplemented from **decaf::net::InetAddress** (p. 1980).

6.394.2.8 virtual bool decaf::net::Inet4Address::isMCSiteLocal ( ) const [virtual]

Check if this **InetAddress** (p. 1974) is Multicast and has Site Local scope.

**Returns**

true if the address is Multicast and has Site Local scope.

Reimplemented from **decaf::net::InetAddress** (p. 1981).

6.394.2.9 `virtual bool decaf::net::Inet4Address::isMulticastAddress ( ) const` [virtual]

Check if this **InetAddress** (p. 1974) is a valid Multicast address.

#### Returns

true if the address is a Multicast address.

Reimplemented from **decaf::net::InetAddress** (p. 1981).

6.394.2.10 `virtual bool decaf::net::Inet4Address::isSiteLocalAddress ( ) const` [virtual]

Check if this **InetAddress** (p. 1974) is a valid site local address.

#### Returns

true if the address is a site local address.

Reimplemented from **decaf::net::InetAddress** (p. 1981).

### 6.394.3 Friends And Related Function Documentation

6.394.3.1 `friend class InetAddress` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet4Address.h`

## 6.395 decaf::net::Inet6Address Class Reference

```
#include <src/main/decaf/net/Inet6Address.h>
```

Inheritance diagram for `decaf::net::Inet6Address`:

#### Public Member Functions

- `virtual ~Inet6Address ()`

#### Protected Member Functions

- `Inet6Address ()`
- `Inet6Address (const unsigned char *ipAddress, int numBytes)`
- `Inet6Address (const std::string &hostname, const unsigned char *ipAddress, int numBytes)`

## Friends

- class **InetAddress**

### 6.395.1 Constructor & Destructor Documentation

6.395.1.1 `decaf::net::Inet6Address::Inet6Address ( )` `[protected]`

6.395.1.2 `decaf::net::Inet6Address::Inet6Address ( const unsigned char * ipAddress, int numBytes )` `[protected]`

6.395.1.3 `decaf::net::Inet6Address::Inet6Address ( const std::string & hostname, const unsigned char * ipAddress, int numBytes )` `[protected]`

6.395.1.4 `virtual decaf::net::Inet6Address::~~Inet6Address ( )` `[virtual]`

### 6.395.2 Friends And Related Function Documentation

6.395.2.1 `friend class InetAddress` `[friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Inet6Address.h`

## 6.396 decaf::net::InetAddress Class Reference

Represents an IP address.

```
#include <src/main/decaf/net/InetAddress.h>
```

Inheritance diagram for `decaf::net::InetAddress`:

## Public Member Functions

- virtual `~InetAddress ( )`
- virtual `decaf::lang::ArrayPointer< unsigned char > getAddress ( ) const`  
*Returns the Raw IP address in Network byte order.*
- virtual `std::string getHostAddress ( ) const`  
*Returns a textual representation of the IP Address.*
- virtual `std::string getHostName ( ) const`  
*Get the host name associated with this **InetAddress** (p. 1974) instance.*
- virtual `std::string toString ( ) const`  
*Returns a string representation of the **InetAddress** (p. 1974) in the form 'hostname / ipaddress'.*

- virtual bool **isAnyLocalAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid wildcard address.*
- virtual bool **isLoopbackAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid loopback address.*
- virtual bool **isMulticastAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid Multicast address.*
- virtual bool **isLinkLocalAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid link local address.*
- virtual bool **isSiteLocalAddress** () const  
*Check if this **InetAddress** (p. 1974) is a valid site local address.*
- virtual bool **isMCGlobal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Global scope.*
- virtual bool **isMCNodeLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Node Local scope.*
- virtual bool **isMCLinkLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Link Local scope.*
- virtual bool **isMCSiteLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Site Local scope.*
- virtual bool **isMCOrgLocal** () const  
*Check if this **InetAddress** (p. 1974) is Multicast and has Organization scope.*

### Static Public Member Functions

- static **InetAddress** **getByAddress** (const unsigned char \*bytes, int numBytes)  
*Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 1974) instance.*
- static **InetAddress** **getByAddress** (const std::string &hostname, const unsigned char \*bytes, int numBytes)  
*Given a host name and IPAddress return a new **InetAddress** (p. 1974).*
- static **InetAddress** **getLocalHost** ()  
*Gets an **InetAddress** (p. 1974) that is the local host address.*

### Protected Member Functions

- **InetAddress** ()
- **InetAddress** (const unsigned char \*ipAddress, int numBytes)
- **InetAddress** (const std::string &hostname, const unsigned char \*ipAddress, int numBytes)

## Static Protected Member Functions

- static unsigned int **bytesToInt** (const unsigned char \*bytes, int start)  
*Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.*
- static **InetAddress** **getAnyAddress** ()
- static **InetAddress** **getLoopbackAddress** ()

## Protected Attributes

- std::string **hostname**
- bool **reached**
- **decaf::lang::ArrayPointer**< unsigned char > **addressBytes**

## Static Protected Attributes

- static const unsigned char **loopbackBytes** [4]
- static const unsigned char **anyBytes** [4]

### 6.396.1 Detailed Description

Represents an IP address.

Since

1.0

### 6.396.2 Constructor & Destructor Documentation

6.396.2.1 **decaf::net::InetAddress::InetAddress** ( ) [protected]

6.396.2.2 **decaf::net::InetAddress::InetAddress** ( const unsigned char \* *ipAddress*, int *numBytes* ) [protected]

6.396.2.3 **decaf::net::InetAddress::InetAddress** ( const std::string & *hostname*, const unsigned char \* *ipAddress*, int *numBytes* ) [protected]

6.396.2.4 **virtual decaf::net::InetAddress::~~InetAddress** ( ) [virtual]

### 6.396.3 Member Function Documentation

6.396.3.1 **static unsigned int decaf::net::InetAddress::bytesToInt** ( const unsigned char \* *bytes*, int *start* ) [static, protected]

Converts the bytes in an address array to an int starting from the value start treating the start value as the high order byte.

**Parameters**

<i>bytes</i>	The array of bytes to convert to an int.
<i>start</i>	The index in the array to treat as the high order byte.

**Returns**

an unsigned int that represents the address value.

6.396.3.2 `virtual decaf::lang::ArrayPointer<unsigned char>  
decaf::net::InetAddress::getAddress ( ) const [virtual]`

Returns the Raw IP address in Network byte order.

The returned address is a copy of the bytes contained in this **InetAddress** (p. 1974).

**Returns**

and **ArrayPointer** containing the raw bytes of the network address.

6.396.3.3 `static InetAddress decaf::net::InetAddress::getAnyAddress ( ) [static,  
protected]`

6.396.3.4 `static InetAddress decaf::net::InetAddress::getByAddress ( const std::string &  
hostname, const unsigned char * bytes, int numBytes ) [static]`

Given a host name and **IPAddress** return a new **InetAddress** (p. 1974).

There is no name service checking or address validation done on the provided host name. The host name can either be machine name or the text based representation of the IP Address.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

**Returns**

a copy of an **InetAddress** (p. 1974) that represents the given byte array address.

**Exceptions**

<b><i>UnknownHostException</i></b> (p. 3841)	if the address array length is invalid.
---	---



6.396.3.5 **static InetAddress** decaf::net::InetAddress::getByAddress ( **const unsigned char \***  
*bytes*, **int numBytes** ) [static]

Given a raw IP Address in byte array form, create and return a new **InetAddress** (p. 1974) instance.

An IPV4 address must be only four bytes in length and an IPV6 address must be 16 bytes in length.

#### Returns

a copy of an **InetAddress** (p. 1974) that represents the given byte array address.

#### Exceptions

<b>UnknownHostException</b> (p. 3841)	if the address array length is invalid.
--	---

6.396.3.6 **virtual std::string** decaf::net::InetAddress::getHostAddress ( ) **const**  
[virtual]

Returns a textual representation of the IP Address.

#### Returns

the string form of the IP Address.

6.396.3.7 **virtual std::string** decaf::net::InetAddress::getHostName ( ) **const** [virtual]

Get the host name associated with this **InetAddress** (p. 1974) instance.

If a host name was set upon construction then that value is returned, otherwise a reverse name lookup will be performed to attempt to get the host name associated with the set IP Address. If the host name cannot be resolved the textual representation of the IP Address is returned instead.

#### Returns

the name of the host associated with this set IP Address.

6.396.3.8 **static InetAddress** decaf::net::InetAddress::getLocalHost ( ) [static]

Gets an **InetAddress** (p. 1974) that is the local host address.

If the localhost value cannot be resolved then the **InetAddress** (p. 1974) for Loopback is returned.

**Returns**

a new **InetAddress** (p. 1974) object that contains the local host address.

**Exceptions**

<b><i>UnknownHostException</i></b> (p. 3841)	if the address for local host is not found.
---	---

6.396.3.9 **static InetAddress decaf::net::InetAddress::getLoopbackAddress ( )**  
[static, protected]

6.396.3.10 **virtual bool decaf::net::InetAddress::isAnyLocalAddress ( ) const** [inline, virtual]

Check if this **InetAddress** (p. 1974) is a valid wildcard address.

**Returns**

true if the address is a wildcard address.

Reimplemented in **decaf::net::Inet4Address** (p. 1971).

6.396.3.11 **virtual bool decaf::net::InetAddress::isLinkLocalAddress ( ) const** [inline, virtual]

Check if this **InetAddress** (p. 1974) is a valid link local address.

**Returns**

true if the address is a link local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1971).

6.396.3.12 **virtual bool decaf::net::InetAddress::isLoopbackAddress ( ) const** [inline, virtual]

Check if this **InetAddress** (p. 1974) is a valid loopback address.

**Returns**

true if the address is a loopback address.

Reimplemented in **decaf::net::Inet4Address** (p. 1972).

6.396.3.13 `virtual bool decaf::net::InetAddress::isMCGlobal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 1974) is Multicast and has Global scope.

#### Returns

true if the address is Multicast and has Global scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1972).

6.396.3.14 `virtual bool decaf::net::InetAddress::isMCLinkLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 1974) is Multicast and has Link Local scope.

#### Returns

true if the address is Multicast and has Link Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1972).

6.396.3.15 `virtual bool decaf::net::InetAddress::isMCNodeLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 1974) is Multicast and has Node Local scope.

#### Returns

true if the address is Multicast and has Node Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1972).

6.396.3.16 `virtual bool decaf::net::InetAddress::isMCOrgLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 1974) is Multicast and has Organization scope.

#### Returns

true if the address is Multicast and has Organization scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1972).

6.396.3.17 `virtual bool decaf::net::InetAddress::isMCSiteLocal ( ) const [inline, virtual]`

Check if this **InetAddress** (p. 1974) is Multicast and has Site Local scope.

**Returns**

true if the address is Multicast and has Site Local scope.

Reimplemented in **decaf::net::Inet4Address** (p. 1973).

```
6.396.3.18  virtual bool decaf::net::InetAddress::isMulticastAddress ( ) const  [inline,
virtual]
```

Check if this **InetAddress** (p. 1974) is a valid Multicast address.

**Returns**

true if the address is a Multicast address.

Reimplemented in **decaf::net::Inet4Address** (p. 1973).

```
6.396.3.19  virtual bool decaf::net::InetAddress::isSiteLocalAddress ( ) const  [inline,
virtual]
```

Check if this **InetAddress** (p. 1974) is a valid site local address.

**Returns**

true if the address is a site local address.

Reimplemented in **decaf::net::Inet4Address** (p. 1973).

```
6.396.3.20  virtual std::string decaf::net::InetAddress::toString ( ) const  [virtual]
```

Returns a string representation of the **InetAddress** (p. 1974) in the form 'hostname / ipaddress'.

If the hostname is not resolved than it appears as empty.

**Returns**

string value of this **InetAddress** (p. 1974).

**6.396.4 Field Documentation**

```
6.396.4.1  decaf::lang::ArrayPointer<unsigned char>
decaf::net::InetAddress::addressBytes  [mutable, protected]
```

```
6.396.4.2  const unsigned char decaf::net::InetAddress::anyBytes[4]  [static,
protected]
```

6.396.4.3 `std::string decaf::net::InetAddress::hostname` [mutable, protected]

6.396.4.4 `const unsigned char decaf::net::InetAddress::loopbackBytes[4]` [static, protected]

6.396.4.5 `bool decaf::net::InetAddress::reached` [mutable, protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/InetAddress.h`

## 6.397 decaf::net::InetSocketAddress Class Reference

```
#include <src/main/decaf/net/InetSocketAddress.h>
```

Inheritance diagram for `decaf::net::InetSocketAddress`:

### Public Member Functions

- `InetSocketAddress ()`
- `virtual ~InetSocketAddress ()`

### 6.397.1 Constructor & Destructor Documentation

6.397.1.1 `decaf::net::InetSocketAddress::InetSocketAddress ( )`

6.397.1.2 `virtual decaf::net::InetSocketAddress::~~InetSocketAddress ( )` [virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/net/InetSocketAddress.h`

## 6.398 inflate\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/inflate.h>
```

### Data Fields

- `inflate_mode mode`
- `int last`

- int **wrap**
- int **havedict**
- int **flags**
- unsigned **dmax**
- unsigned long **check**
- unsigned long **total**
- **gz\_headerp** **head**
- unsigned **wbits**
- unsigned **wsize**
- unsigned **whave**
- unsigned **wnext**
- unsigned char FAR \* **window**
- unsigned long **hold**
- unsigned **bits**
- unsigned **length**
- unsigned **offset**
- unsigned **extra**
- **code** const FAR \* **lencode**
- **code** const FAR \* **distcode**
- unsigned **lenbits**
- unsigned **distbits**
- unsigned **ncode**
- unsigned **nlen**
- unsigned **ndist**
- unsigned **have**
- **code** FAR \* **next**
- unsigned short **lens** [320]
- unsigned short **work** [288]
- **code** **codes** [ENOUGH]
- int **sane**
- int **back**
- unsigned **was**

### 6.398.1 Field Documentation

6.398.1.1 int **inflate\_state::back**

6.398.1.2 unsigned **inflate\_state::bits**

6.398.1.3 unsigned long **inflate\_state::check**

6.398.1.4 **code** **inflate\_state::codes**[ENOUGH]

6.398.1.5 unsigned **inflate\_state::distbits**

6.398.1.6 **code** const FAR\* **inflate\_state::distcode**

- 6.398.1.7 unsigned inflate\_state::dmax
- 6.398.1.8 unsigned inflate\_state::extra
- 6.398.1.9 int inflate\_state::flags
- 6.398.1.10 unsigned inflate\_state::have
- 6.398.1.11 int inflate\_state::havedict
- 6.398.1.12 gz\_headerp inflate\_state::head
- 6.398.1.13 unsigned long inflate\_state::hold
- 6.398.1.14 int inflate\_state::last
- 6.398.1.15 unsigned inflate\_state::lenbits
- 6.398.1.16 code const FAR\* inflate\_state::lencode
- 6.398.1.17 unsigned inflate\_state::length
- 6.398.1.18 unsigned short inflate\_state::lens[320]
- 6.398.1.19 inflate\_mode inflate\_state::mode
- 6.398.1.20 unsigned inflate\_state::ncode
- 6.398.1.21 unsigned inflate\_state::ndist
- 6.398.1.22 code FAR\* inflate\_state::next
- 6.398.1.23 unsigned inflate\_state::nlen
- 6.398.1.24 unsigned inflate\_state::offset
- 6.398.1.25 int inflate\_state::sane
- 6.398.1.26 unsigned long inflate\_state::total
- 6.398.1.27 unsigned inflate\_state::was
- 6.398.1.28 unsigned inflate\_state::wbits
- 6.398.1.29 unsigned inflate\_state::whave
- 6.398.1.30 unsigned char FAR\* inflate\_state::window

6.398.1.31 unsigned inflate\_state::wnext

6.398.1.32 unsigned short inflate\_state::work[288]

6.398.1.33 int inflate\_state::wrap

6.398.1.34 unsigned inflate\_state::wsiz

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/inflate.h

## 6.399 decaf::util::zip::Inflater Class Reference

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see *specification*).

```
#include <src/main/decaf/util/zip/Inflater.h>
```

### Public Member Functions

- **Inflater** ()

*Creates a new decompressor.*

- **Inflater** (bool nowrap)

*Creates a new decompressor.*

- virtual ~**Inflater** ()

- void **setInput** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

*Sets input data for decompression.*

- void **setInput** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

*Sets input data for decompression.*

- void **setInput** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException )

*Sets input data for decompression.*

- int **getRemaining** () const

*Returns the total number of bytes remaining in the input buffer.*

- void **setDictionary** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

*Sets the preset dictionary to the given array of bytes.*



- void **setDictionary** (const std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Sets the preset dictionary to the given array of bytes.*
- void **setDictionary** (const std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the preset dictionary to the given array of bytes.*
- bool **needsInput** () const
- bool **needsDictionary** () const
- void **finish** ()  
*When called, indicates that decompression should end with the current contents of the input buffer.*
- bool **finished** () const
- int **inflate** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )  
*Uncompresses bytes into specified buffer.*
- int **inflate** (std::vector< unsigned char > &buffer, int offset, int length) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )  
*Uncompresses bytes into specified buffer.*
- int **inflate** (std::vector< unsigned char > &buffer) throw ( decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException )  
*Uncompresses bytes into specified buffer.*
- long long **getAdler** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesRead** () const throw ( decaf::lang::exceptions::IllegalStateException )
- long long **getBytesWritten** () const throw ( decaf::lang::exceptions::IllegalStateException )
- void **reset** () throw ( decaf::lang::exceptions::IllegalStateException )  
*Resets inflater so that a new set of input data can be processed.*
- void **end** ()  
*Closes the decompressor and discards any unprocessed input.*

### 6.399.1 Detailed Description

This class uncompresses data that was compressed using the *DEFLATE* algorithm (see *specification*).

Basically this class is part of the API to the stream based ZLIB compression library and is used as such by **InflaterInputStream** (p. 1994) and its descendants.

The typical usage of a **Inflater** (p. 1985) outside this package consists of a specific call to one of its constructors before being passed to an instance of **InflaterInputStream** (p. 1994).

**See also****InflaterInputStream** (p. 1994)**Deflater** (p. 1672)**Since**

1.0

**6.399.2 Constructor & Destructor Documentation****6.399.2.1** `decaf::util::zip::Inflater::Inflater ( )`

Creates a new decompressor.

This constructor defaults the inflater to use the ZLIB header and checksum fields.

**6.399.2.2** `decaf::util::zip::Inflater::Inflater ( bool nowrap )`

Creates a new decompressor.

If the parameter 'nowrap' is true then the ZLIB header and checksum fields will not be used. This provides compatibility with the compression format used by both GZIP and PKZIP.

Note: When using the 'nowrap' option it is also necessary to provide an extra "dummy" byte as input. This is required by the ZLIB native library in order to support certain optimizations.

**6.399.2.3** `virtual decaf::util::zip::Inflater::~~Inflater ( )` [virtual]**6.399.3 Member Function Documentation****6.399.3.1** `void decaf::util::zip::Inflater::end ( )`

Closes the decompressor and discards any unprocessed input.

This method should be called when the decompressor is no longer being used, but will also be called automatically by the destructor. Once this method is called, the behavior of the **Inflater** (p. 1985) object is undefined.

**6.399.3.2** `void decaf::util::zip::Inflater::finish ( )`

When called, indicates that decompression should end with the current contents of the input buffer.

6.399.3.3 `bool decaf::util::zip::Inflater::finished ( ) const`

#### Returns

true if the end of the compressed data output stream has been reached.

6.399.3.4 `long long decaf::util::zip::Inflater::getAdler ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the ADLER-32 value of the uncompressed data.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.399.3.5 `long long decaf::util::zip::Inflater::getBytesRead ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of compressed bytes input so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.399.3.6 `long long decaf::util::zip::Inflater::getBytesWritten ( ) const throw ( decaf::lang::exceptions::IllegalStateException )`

#### Returns

the total number of decompressed bytes output so far.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.399.3.7 `int decaf::util::zip::Inflater::getRemaining ( ) const`

Returns the total number of bytes remaining in the input buffer.

This can be used to find out what bytes still remain in the input buffer after decompression has finished.

**Returns**

the total number of bytes remaining in the input buffer

6.399.3.8 `int decaf::util::zip::Inflater::inflate ( std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException, decaf::util::zip::DataFormatException )`

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1991) or **needsDictionary()** (p. 1990) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1988) can be used to get the Adler-32 value of the dictionary required.

**Parameters**

<i>buffer</i>	The Buffer to write the compressed data to.
---------------	---

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
<b>DataFormatException</b> (p. 1520)	if the compressed data format is invalid.

6.399.3.9 `int decaf::util::zip::Inflater::inflate ( std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )`

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1991) or **needsDictionary()** (p. 1990) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1988) can be used to get the Adler-32 value of the dictionary required.

**Parameters**

<i>buffer</i>	The Buffer to write the compressed data to.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.

<b>DataFormatException</b> (p. 1520)	if the compressed data format is invalid.
---	---

6.399.3.10 int decaf::util::zip::Inflater::inflate ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::util::zip::DataFormatException )

Uncompresses bytes into specified buffer.

Returns actual number of bytes uncompressed. A return value of 0 indicates that **needsInput()** (p. 1991) or **needsDictionary()** (p. 1990) should be called in order to determine if more input data or a preset dictionary is required. In the latter case, **getAdler()** (p. 1988) can be used to get the Adler-32 value of the dictionary required.

#### Parameters

<i>buffer</i>	The Buffer to write the compressed data to.
<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start writing at.
<i>length</i>	The maximum number of byte of data to write.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL.
<i>IllegalStateException</i>	if in the end state.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<b>DataFormatException</b> (p. 1520)	if the compressed data format is invalid.

6.399.3.11 bool decaf::util::zip::Inflater::needsDictionary ( ) const

#### Returns

true if a preset dictionary is needed for decompression.

6.399.3.12 bool decaf::util::zip::Inflater::needsInput ( ) const

#### Returns

true if the input data buffer is empty and **setInput()** (p. 1993) should be called in order to provide more input

6.399.3.13 `void decaf::util::zip::Inflater::reset ( ) throw ( decaf::lang::exceptions::IllegalStateException )`

Resets deflater so that a new set of input data can be processed.

Keeps current decompression level and strategy settings.

#### Exceptions

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------

6.399.3.14 `void decaf::util::zip::Inflater::setDictionary ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1990) returns 0 and **needsDictionary()** (p. 1990) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1988) can be used to get the Adler-32 value of the dictionary needed.

#### Parameters

<i>buffer</i>	The Buffer to read in for decompression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgumentException</i>	if the given dictionary doesn't match the required dictionaries checksum value.

6.399.3.15 `void decaf::util::zip::Inflater::setDictionary ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )`

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1990) returns 0 and **needsDictionary()** (p. 1990) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1988) can be used to get the Adler-32 value of the dictionary needed.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
<i>size</i>	The size of the buffer passed in.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgumentException</i>	if the given dictionary doesn't match the required dictionaries checksum value.

6.399.3.16 void decaf::util::zip::Inflater::setDictionary ( const std::vector< unsigned char > & *buffer* ) throw ( decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::IllegalArgumentException )

Sets the preset dictionary to the given array of bytes.

Should be called when **inflate()** (p. 1990) returns 0 and **needsDictionary()** (p. 1990) returns true indicating that a preset dictionary is required. The method **getAdler()** (p. 1988) can be used to get the Adler-32 value of the dictionary needed.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
<i>IllegalArgumentException</i>	if the given dictionary doesn't match the required dictionaries checksum value.

6.399.3.17 void decaf::util::zip::Inflater::setInput ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1991) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
<i>size</i>	The size of the buffer passed in.

<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.399.3.18 `void decaf::util::zip::Inflater::setInput ( const std::vector< unsigned char > & buffer, int offset, int length ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::IllegalStateException )`

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1991) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
<i>offset</i>	The position in the Buffer to start reading from.
<i>length</i>	The number of bytes to read from the input buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the offset + length > size of the buffer.
<i>IllegalStateException</i>	if in the end state.

6.399.3.19 `void decaf::util::zip::Inflater::setInput ( const std::vector< unsigned char > & buffer ) throw ( decaf::lang::exceptions::IllegalStateException )`

Sets input data for decompression.

This should be called whenever **needsInput()** (p. 1991) returns true indicating that more input data is required.

**Parameters**

<i>buffer</i>	The Buffer to read in for decompression.
---------------	--

**Exceptions**

<i>IllegalStateException</i>	if in the end state.
------------------------------	----------------------



The documentation for this class was generated from the following file:

- src/main/decaf/util/zip/**Inflater.h**

## 6.400 decaf::util::zip::InflaterInputStream Class Reference

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

```
#include <src/main/decaf/util/zip/InflaterInputStream.h>
```

Inheritance diagram for decaf::util::zip::InflaterInputStream:

### Public Member Functions

- **InflaterInputStream** (decaf::io::InputStream \*inputStream, bool own=false)  
*Create an instance of this class with a default inflater and buffer size.*
- **InflaterInputStream** (decaf::io::InputStream \*inputStream, Inflater \*inflater, bool own=false)  
*Creates a new **InflaterInputStream** (p. 1994) with a user supplied **Inflater** (p. 1985) and a default buffer size.*
- **InflaterInputStream** (decaf::io::InputStream \*inputStream, Inflater \*inflater, int bufferSize, bool own=false)  
*Creates a new **InflaterInputStream** with a user supplied **Inflater** (p. 1985) and specified buffer size.*
- virtual ~**InflaterInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

#### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.*  
*The default implementation of this method does nothing.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Skips over and discards n bytes of data from this input stream.*

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

num	The number of bytes to skip.
-----	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<b>UnsupportedOperationException</b>	if the concrete stream class does not support skipping bytes.

- virtual void **mark** (int readLimit)

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

readLimit	The max bytes read before marked position is invalid.
-----------	---

- virtual void **reset** () throw ( decaf::io::IOException )

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

*Whether or not mark and reset are supported is an invariant property of a particular input stream instance.*

*The default implementation of this method returns false.*

**Returns**

*true if this stream instance supports marks*

**Protected Member Functions**

- virtual void **fill** () throw ( decaf::io::IOException )  
*Fills the input buffer with the next chunk of data.*
- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int **length**) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**Protected Attributes**

- **Inflater \* inflater**  
*The **Inflater** (p. 1985) instance to use.*
- std::vector< unsigned char > **buff**  
*The buffer to hold chunks of data read from the stream before inflation.*
- int **length**  
*The amount of data currently stored in the input buffer.*
- bool **ownInflater**
- bool **atEOF**

**Static Protected Attributes**

- static const int **DEFAULT\_BUFFER\_SIZE**

**6.400.1 Detailed Description**

A FilterInputStream that decompresses data read from the wrapped InputStream instance.

**Since**

1.0

**6.400.2 Constructor & Destructor Documentation**

- 6.400.2.1 decaf::util::zip::InflaterInputStream::InflaterInputStream ( decaf::io::InputStream \* *inputStream*, bool *own* = false )

Create an instance of this class with a default inflater and buffer size.

**Parameters**

<i>inputStream</i>	The <code>InputStream</code> instance to wrap.
<i>own</i>	Should this <code>Filter</code> take ownership of the <code>InputStream</code> pointer (defaults to false).

6.400.2.2 `decaf::util::zip::InflaterInputStream::InflaterInputStream ( decaf::io::InputStream * inputStream, Inflater * inflater, bool own = false )`

Creates a new **InflaterInputStream** (p. 1994) with a user supplied **Inflater** (p. 1985) and a default buffer size.

When the user supplied a **Inflater** (p. 1985) instance the **InflaterInputStream** (p. 1994) does not take ownership of the **Inflater** (p. 1985) pointer, the caller is still responsible for deleting the **Inflater** (p. 1985).

**Parameters**

<i>inputStream</i>	The <code>InputStream</code> instance to wrap.
<i>inflater</i>	The user supplied <b>Inflater</b> (p. 1985) to use for decompression. (
<i>own</i>	Should this filter take ownership of the <code>InputStream</code> pointer (default is false).

**Exceptions**

<i>NullPointerException</i>	if the <b>Inflater</b> (p. 1985) given is NULL.
-----------------------------	---

6.400.2.3 `decaf::util::zip::InflaterInputStream::InflaterInputStream ( decaf::io::InputStream * inputStream, Inflater * inflater, int bufferSize, bool own = false )`

Creates a new `DeflateOutputStream` with a user supplied **Inflater** (p. 1985) and specified buffer size.

When the user supplied a **Inflater** (p. 1985) instance the **InflaterInputStream** (p. 1994) does not take ownership of the **Inflater** (p. 1985) pointer, the caller is still responsible for deleting the **Inflater** (p. 1985).

**Parameters**

<i>inputStream</i>	The <code>InputStream</code> instance to wrap.
<i>inflater</i>	The user supplied <b>Inflater</b> (p. 1985) to use for decompression.
<i>bufferSize</i>	The size of the input buffer.
<i>own</i>	Should this filter take ownership of the <code>InputStream</code> pointer (default is false).

**Exceptions**

<i>NullPointerException</i>	if the <b>Inflater</b> (p. 1985) given is NULL.
<i>IllegalArgumentException</i>	if the <code>bufferSize</code> value is zero.

6.400.2.4 virtual decaf::util::zip::InflaterInputStream::~~InflaterInputStream ( ) [virtual]

### 6.400.3 Member Function Documentation

6.400.3.1 virtual int decaf::util::zip::InflaterInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Until EOF this method always returns 1, thereafter it always returns 0.

Reimplemented from **decaf::io::FilterInputStream** (p. 1857).

6.400.3.2 virtual void decaf::util::zip::InflaterInputStream::close ( ) throw ( decaf::io::IOException ) [virtual]

Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Closes any resources associated with this **InflaterInputStream** (p. 1994).

Reimplemented from **decaf::io::FilterInputStream** (p. 1857).

6.400.3.3 virtual int decaf::util::zip::InflaterInputStream::doReadArrayBounded ( unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.400.3.4 `virtual int decaf::util::zip::InflaterInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.400.3.5 `virtual void decaf::util::zip::InflaterInputStream::fill ( ) throw ( decaf::io::IOException )` [protected, virtual]

Fills the input buffer with the next chunk of data.

### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

6.400.3.6 `virtual void decaf::util::zip::InflaterInputStream::mark ( int readLimit )` [virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Does nothing.

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.400.3.7 `virtual bool decaf::util::zip::InflaterInputStream::markSupported ( ) const` [virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

### Returns

true if this stream instance supports marks

Always returns false.

Reimplemented from **decaf::io::FilterInputStream** (p. 1859).

6.400.3.8 `virtual void decaf::util::zip::InflaterInputStream::reset ( ) throw ( decaf::io::IOException ) [virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: \* If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`.

If the method `markSupported` returns false, then: \* The call to `reset` may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Always throws an **IOException** when called.

Reimplemented from **decaf::io::FilterInputStream** (p. 1859).

6.400.3.9 `virtual long long decaf::util::zip::InflaterInputStream::skip ( long long num ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Skips over and discards `n` bytes of data from this input stream.

The `skip` method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before `n` bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The `skip` method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until `num` bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters**

<i>num</i>	The number of bytes to skip.
------------	------------------------------

**Returns**

total bytes skipped

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<b><i>UnsupportedOperationException</i></b>	if the concrete stream class does not support skipping bytes.

Skips the specified amount of uncompressed input data.

Reimplemented from **decaf::io::FilterInputStream** (p. 1860).

**6.400.4 Field Documentation**

**6.400.4.1** `bool decaf::util::zip::InflaterInputStream::atEOF` [protected]

**6.400.4.2** `std::vector<unsigned char> decaf::util::zip::InflaterInputStream::buff`  
[protected]

The buffer to hold chunks of data read from the stream before inflation.

**6.400.4.3** `const int decaf::util::zip::InflaterInputStream::DEFAULT_BUFFER_SIZE`  
[static, protected]

**6.400.4.4** `Inflater* decaf::util::zip::InflaterInputStream::inflater` [protected]

The **Inflater** (p. 1985) instance to use.

**6.400.4.5** `int decaf::util::zip::InflaterInputStream::length` [protected]

The amount of data currently stored in the input buffer.

**6.400.4.6** `bool decaf::util::zip::InflaterInputStream::ownInflater` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/InflaterInputStream.h`



## 6.401 decaf::io::InputStream Class Reference

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

```
#include <src/main/decaf/io/InputStream.h>
```

Inheritance diagram for decaf::io::InputStream:

### Public Member Functions

- **InputStream** ()
- virtual **~InputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **InputStream** (p. 2002) freeing any resources that might have been aquired during the lifetime of this stream.*
- virtual void **mark** (int readLimit)  
*Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Repositions this stream to the position at the time the mark method was last called on this input stream.*
- virtual bool **markSupported** () const  
*Determines if this input stream supports the mark and reset methods.*
- virtual int **available** () const throw ( decaf::io::IOException )  
*Indicates the number of bytes available.*
- virtual int **read** () throw ( decaf::io::IOException )  
*Reads a single byte from the buffer.*
- virtual int **read** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Reads up to size bytes of data from the input stream into an array of bytes.*
- virtual int **read** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Reads up to length bytes of data from the input stream into an array of bytes.*
- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Skips over and discards n bytes of data from this input stream.*
- virtual std::string **toString** () const  
*Output a String representation of this object.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

## Protected Member Functions

- virtual int **doReadByte** ()=0 throw ( decaf::io::IOException )
- virtual int **doReadArray** (unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

### 6.401.1 Detailed Description

A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.

#### Since

1.0

### 6.401.2 Constructor & Destructor Documentation

#### 6.401.2.1 decaf::io::InputStream::InputStream ( )

6.401.2.2 virtual decaf::io::InputStream::~~InputStream ( ) [virtual]

### 6.401.3 Member Function Documentation

6.401.3.1 virtual int decaf::io::InputStream::available ( ) const throw ( decaf::io::IOException ) [inline, virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented in **decaf::internal::io::StandardInputStream** (p. 3524), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2833), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3692), **decaf::io::BlockingByteArrayInputStream** (p. 802), **decaf::io::BufferedInputStream** (p. 896), **decaf::io::ByteArrayInputStream** (p. 988), **decaf::io::FilterInputStream** (p. 1857), **decaf::io::PushbackInputStream** (p. 3089), and **decaf::util::zip::InflaterInputStream** (p. 1998).

6.401.3.2 virtual void decaf::io::InputStream::close ( ) throw ( decaf::io::IOException ) [virtual]

Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Implements **decaf::io::Closeable** (p. 1121).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2834), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3693), **decaf::io::BlockingByteArrayInputStream** (p. 802), **decaf::io::BufferedInputStream** (p. 897), **decaf::io::FilterInputStream** (p. 1857), and **decaf::util::zip::InflaterInputStream** (p. 1998).

6.401.3.3 `virtual int decaf::io::InputStream::doReadArray ( unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented in **decaf::io::FilterInputStream** (p. 1857).

6.401.3.4 `virtual int decaf::io::InputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException ) [protected, virtual]`

Reimplemented in **activemq::io::LoggingInputStream** (p. 2359), **decaf::internal::net::ssl::openssl::OpenSSLInputStream** (p. 2834), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3693), **decaf::io::BlockingByteArrayInputStream** (p. 803), **decaf::io::BufferedInputStream** (p. 897), **decaf::io::ByteArrayInputStream** (p. 989), **decaf::io::FilterInputStream** (p. 1858), **decaf::io::PushbackInputStream** (p. 3090), **decaf::util::zip::CheckedInputStream** (p. 1111), and **decaf::util::zip::InflaterInputStream** (p. 1999).

6.401.3.5 `virtual int decaf::io::InputStream::doReadByte ( ) throw ( decaf::io::IOException ) [protected, pure virtual]`

Implemented in **activemq::io::LoggingInputStream** (p. 2359), **decaf::internal::io::StandardInputStream** (p. 3525), **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2834), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3693), **decaf::io::BlockingByteArrayInputStream** (p. 803), **decaf::io::BufferedInputStream** (p. 897), **decaf::io::ByteArrayInputStream** (p. 989), **decaf::io::FilterInputStream** (p. 1858), **decaf::io::PushbackInputStream** (p. 3090), **decaf::util::zip::CheckedInputStream** (p. 1111), and **decaf::util::zip::InflaterInputStream** (p. 1999).

6.401.3.6 `virtual void decaf::io::InputStream::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Locks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

6.401.3.7 `virtual void decaf::io::InputStream::mark ( int readLimit ) [virtual]`

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same

bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Calling mark on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented in **decaf::io::BufferedInputStream** (p. 897), **decaf::io::ByteArrayInputStream** (p. 989), **decaf::io::FilterInputStream** (p. 1858), **decaf::io::PushbackInputStream** (p. 3090), and **decaf::util::zip::InflaterInputStream** (p. 1999).

```
6.401.3.8 virtual bool decaf::io::InputStream::markSupported ( ) const [inline,
virtual]
```

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

#### Returns

true if this stream instance supports marks

Reimplemented in **decaf::io::BufferedInputStream** (p. 897), **decaf::io::ByteArrayInputStream** (p. 989), **decaf::io::FilterInputStream** (p. 1859), **decaf::io::PushbackInputStream** (p. 3090), and **decaf::util::zip::InflaterInputStream** (p. 2000).

```
6.401.3.9 virtual void decaf::io::InputStream::notify ( ) throw
( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
virtual]
```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

```
6.401.3.10 virtual void decaf::io::InputStream::notifyAll (    ) throw
( decaf::lang::exceptions::RuntimeException,
  decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
  virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

```
6.401.3.11 virtual int decaf::io::InputStream::read ( unsigned char * buffer, int
  size, int offset, int length ) throw ( decaf::io::IOException,
  decaf::lang::exceptions::IndexOutOfBoundsException,
  decaf::lang::exceptions::NullPointerException ) [virtual]
```

Reads up to length bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as length bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If length is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

The first byte read is stored into element b[offset], the next one into b[offset+1], and so on. The number of bytes read is, at most, equal to length. Let k be the number of bytes actually read; these bytes will be stored in elements b[offset] through b[offset+k-1], leaving elements b[offset+k] through b[offset+length-1] unaffected.

In every case, elements b[0] through b[offset] and elements b[offset+length] through b[b.length-1] are unaffected.

This method called the protected virtual method doReadArrayBounded which simply calls the method **doReadByte()** (p. 2005) repeatedly. If the first such call results in an **IOException** (p. 2103), that exception is returned. If any subsequent call to **doReadByte()** (p. 2005) results in a **IOException** (p. 2103), the exception is caught and treated as if it were end of file; the bytes read up to that point are stored into the buffer and the number of bytes read before the exception occurred is returned. The default implementation of this method blocks until the requested amount of input data has been read, end of file is detected, or an exception is thrown. Subclasses are encouraged to provide a more efficient implementation of this method.

**Parameters**

<i>buffer</i>	The target buffer to write the read in data to.
<i>size</i>	The size in bytes of the target buffer.
<i>offset</i>	The position in the buffer to start inserting the read in data.
<i>length</i>	The maximum number of bytes that should be read into buffer.

**Returns**

The number of bytes read or -1 if EOF is detected

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if length > size - offset.

6.401.3.12 `virtual int decaf::io::InputStream::read ( ) throw ( decaf::io::IOException )`  
[virtual]

Reads a single byte from the buffer.

The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

The default implementation of this method calls the internal virtual method `doReadByte` which is a pure virtual method and must be overridden by all subclasses.

**Returns**

The next byte or -1 if the end of stream is reached.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

6.401.3.13 `virtual int decaf::io::InputStream::read ( unsigned char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )`  
[virtual]

Reads up to size bytes of data from the input stream into an array of bytes.

An attempt is made to read as many as size bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If size is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

This method called the protected virtual method `doReadArray` which by default is the same as calling `read( buffer, size, 0, size )`. Subclasses can customize the behavior of this method by overriding the `doReadArray` method to provide a better performing read operation.

#### Parameters

<i>buffer</i>	The target buffer to write the read in data to.
<i>size</i>	The size in bytes of the target buffer.

#### Returns

The number of bytes read or -1 if EOF is detected

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<b><i>NullPointerException</i></b>	if buffer passed is NULL.

6.401.3.14 `virtual void decaf::io::InputStream::reset ( ) throw ( decaf::io::IOException )`  
[virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: \* If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method `markSupported` returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------



Reimplemented in **decaf::io::BufferedInputStream** (p. 898), **decaf::io::ByteArrayInputStream** (p. 990), **decaf::io::FilterInputStream** (p. 1859), **decaf::io::PushbackInputStream** (p. 3091), and **decaf::util::zip::InflaterInputStream** (p. 2000).

```
6.401.3.15  virtual long long decaf::io::InputStream::skip ( long
              long num ) throw ( decaf::io::IOException,
              decaf::lang::exceptions::UnsupportedOperationException )
              [virtual]
```

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<b><i>UnsupportedOperationException</i></b>	if the concrete stream class does not support skipping bytes.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream** (p. 2834), **decaf::internal::net::tcp::TcpSocketInputStream** (p. 3693), **decaf::io::BlockingByteArrayInputStream** (p. 803), **decaf::io::BufferedInputStream** (p. 898), **decaf::io::ByteArrayInputStream** (p. 991), **decaf::io::FilterInputStream** (p. 1860), **decaf::io::PushbackInputStream** (p. 3091), **decaf::util::zip::CheckedInputStream** (p. 1112), and **decaf::util::zip::InflaterInputStream** (p. 2001).

```
6.401.3.16  virtual std::string decaf::io::InputStream::toString ( ) const [virtual]
```

Output a String representation of this object.

The default version of this method just prints the Class Name.

#### Returns

a string representation of the object.

6.401.3.17 `virtual bool decaf::io::InputStream::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

6.401.3.18 `virtual void decaf::io::InputStream::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

6.401.3.19 `virtual void decaf::io::InputStream::wait ( long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

```
6.401.3.20 virtual void decaf::io::InputStream::wait ( ) throw (
    decaf::lang::exceptions::RuntimeException,
    decaf::lang::exceptions::IllegalMonitorStateException,
    decaf::lang::exceptions::InterruptedException ) [inline,
    virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

```
6.401.3.21 virtual void decaf::io::InputStream::wait ( long long millisecs, int
    nanos ) throw ( decaf::lang::exceptions::RuntimeException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalMonitorStateException,
    decaf::lang::exceptions::InterruptedException ) [inline,
    virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InputStream.h`

## 6.402 decaf::io::InputStreamReader Class Reference

An **InputStreamReader** (p. 2013) is a bridge from byte streams to character streams.

```
#include <src/main/decaf/io/InputStreamReader.h>
```

Inheritance diagram for `decaf::io::InputStreamReader`:

### Public Member Functions

- **InputStreamReader** (**InputStream** \*stream, bool own=false)  
*Create a new **InputStreamReader** (p. 2013) that wraps the given **InputStream** (p. 2002).*
- virtual **~InputStreamReader** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*
- virtual bool **ready** () const throw ( decaf::io::IOException )  
*Tells whether this stream is ready to be read.*

### Protected Member Functions

- virtual int **doReadArrayBounded** (char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Override this method to customize the functionality of the method `read( unsigned char* buffer, int size, int offset, int length )`.*
- virtual void **checkClosed** () const throw ( decaf::io::IOException )

#### 6.402.1 Detailed Description

An **InputStreamReader** (p. 2013) is a bridge from byte streams to character streams.

For top efficiency, consider wrapping an **InputStreamReader** (p. 2013) within a **BufferedReader**. For example:

```
BufferedReader* in = new BufferedReader( new InputStreamReader (p. 2013)( System.in, false ), true );
```

#### See also

**OutputStreamWriter** (p. 2864)

Since

1.0

## 6.402.2 Constructor & Destructor Documentation

6.402.2.1 `decaf::io::InputStreamReader::InputStreamReader ( InputStream * stream, bool own = false )`

Create a new **InputStreamReader** (p. 2013) that wraps the given **InputStream** (p. 2002).

### Parameters

<i>stream</i>	The <b>InputStream</b> (p. 2002) to read from. (cannot be null).
<i>own</i>	Does this object own the passed <b>InputStream</b> (p. 2002) (defaults to false).

### Exceptions

<i>NullPointerException</i>	if the passed <b>InputStream</b> (p. 2002) is NULL.
-----------------------------	---

6.402.2.2 `virtual decaf::io::InputStreamReader::~~InputStreamReader ( ) [virtual]`

## 6.402.3 Member Function Documentation

6.402.3.1 `virtual void decaf::io::InputStreamReader::checkClosed ( ) const throw ( decaf::io::IOException ) [protected, virtual]`

6.402.3.2 `virtual void decaf::io::InputStreamReader::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
---------------------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 1121).

6.402.3.3 `virtual int decaf::io::InputStreamReader::doReadArrayBounded ( char *  
buffer, int size, int offset, int length ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException ) [protected,  
virtual]`

Override this method to customize the functionality of the method `read( unsigned char*  
buffer, int size, int offset, int length )`.

All subclasses must override this method to provide the basic **Reader** (p. 3108) functionality.

Implements **decaf::io::Reader** (p. 3110).

6.402.3.4 `virtual bool decaf::io::InputStreamReader::ready ( ) const throw ( decaf::io::IOException ) [virtual]`

Tells whether this stream is ready to be read.

#### Returns

True if the next **read()** (p. 3112) is guaranteed not to block for input, false otherwise.  
Note that returning false does not guarantee that the next read will block.

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::Reader** (p. 3113).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InputStreamReader.h`

## 6.403 decaf::internal::nio::IntArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/IntArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::IntArrayBuffer`:

#### Public Member Functions

- **IntArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **IntArrayBuffer** (p. 2015) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **IntArrayBuffer** (int \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a **IntArrayBuffer** (p. 2015) object that wraps the given array.*

- **IntArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*

- **IntArrayBuffer** (const **IntArrayBuffer** &other)

*Create a **IntArrayBuffer** (p. 2015) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*

- virtual ~**IntArrayBuffer** ()
- virtual int \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the int array that backs this buffer (optional operation).*

*Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

#### Returns

*the array that backs this **Buffer** (p. 887).*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this <b>Buffer</b> (p. 887) is read only.</i>
UnsupportedOperationException	<i>if the underlying store has no array.</i>

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*

*Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*

#### Returns

*The offset into the backing array where index zero starts.*

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this <b>Buffer</b> (p. 887) is read only.</i>
UnsupportedOperationException	<i>if the underlying store has no array.</i>

- virtual **IntBuffer** \* **asReadOnlyBuffer** () const

*Creates a new, read-only int buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will*

not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only int buffer which the caller then owns.

- virtual IntBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **IntBuffer** (p. 2026)

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

- virtual IntBuffer \* **duplicate** ()

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new int **Buffer** (p. 887) which the caller owns.

- virtual int **get** () throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the int at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

- virtual int **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the value at the given index.



**Parameters**

index	The index in the <b>Buffer</b> (p. 887) where the int is to be read.
-------	--

**Returns**

the int that is located at the given index.

**Exceptions**

IndexOutOfBoundsException	if index is not smaller than the buffer's limit, or index is negative.
---------------------------	--

- virtual bool **hasArray** () const

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

- virtual bool **isReadOnly** () const

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

- virtual IntBuffer & **put** (int value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

Writes the given integer into this buffer at the current position, and then increments the position.

**Parameters**

value	The integer value to be written.
-------	----------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

- virtual IntBuffer & **put** (int index, int value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

Writes the given ints into this buffer at the given index.

**Parameters**

index	The position in the <b>Buffer</b> (p. 887) to write the data.
value	The ints to write.

**Returns**

a reference to this buffer.

**Exceptions**

IndexOutOfBoundsException	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
---------------------------	--

<b>ReadOnlyBufferException</b> (p. 3115)	- If this buffer is read-only.
---	--------------------------------

- virtual `IntBuffer * slice ()` const

Creates a new **IntBuffer** (p. 2026) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **IntBuffer** (p. 2026) which the caller owns.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **IntArrayBuffer** (p. 2015) as Read-Only.

### 6.403.1 Constructor & Destructor Documentation

6.403.1.1 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **IntArrayBuffer** (p. 2015) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

6.403.1.2 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( int * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **IntArrayBuffer** (p. 2015) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.403.1.3 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **IntArrayBuffer** (p. 2015) will be that of the remaining capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.403.1.4 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer ( const IntArrayBuffer & other )`

Create a **IntArrayBuffer** (p. 2015) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>IntArrayBuffer</b> (p. 2015) this one is to mirror.
--------------	--

6.403.1.5 `virtual decaf::internal::nio::IntArrayBuffer::~~IntArrayBuffer ( ) [virtual]`

## 6.403.2 Member Function Documentation

6.403.2.1 `virtual int* decaf::internal::nio::IntArrayBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 887).

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 2029).

6.403.2.2 `virtual int decaf::internal::nio::IntArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 2030).

6.403.2.3 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer ( ) const`  
[virtual]

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 2030).

6.403.2.4 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException )` [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **IntBuffer** (p. 2026)

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::IntBuffer** (p. 2030).

6.403.2.5 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::duplicate ( )` [virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new int **Buffer** (p. 887) which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 2031).

6.403.2.6 `virtual int decaf::internal::nio::IntArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException )` [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the int at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return.
--	----------------------------------

Implements **decaf::nio::IntBuffer** (p. 2033).

6.403.2.7 `virtual int decaf::internal::nio::IntArrayBuffer::get ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException )` [virtual]

Absolute get method.

Reads the value at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the int is to be read.
--------------	--

#### Returns

the int that is located at the given index.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::IntBuffer** (p. 2032).

**6.403.2.8** `virtual bool decaf::internal::nio::IntArrayBuffer::hasArray ( ) const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::IntBuffer** (p. 2033).

**6.403.2.9** `virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly ( ) const [inline, virtual]`

Tells whether or not this buffer is read-only.

### Returns

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 890).

**6.403.2.10** `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put ( int index, int value ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given ints into this buffer at the given index.

### Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The ints to write.

### Returns

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	- If this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 2034).

```
6.403.2.11 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put ( int
value ) throw ( decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException ) [virtual]
```

Writes the given integer into this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The integer value to be written.
--------------	----------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::IntBuffer** (p. 2034).

```
6.403.2.12 virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly ( bool value )
[inline, protected, virtual]
```

Sets this **IntArrayBuffer** (p. 2015) as Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---



6.403.2.13 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **IntBuffer** (p.2026) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **IntBuffer** (p.2026) which the caller owns.

Implements **decaf::nio::IntBuffer** (p.2037).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/IntArrayBuffer.h`

## 6.404 decaf::nio::IntBuffer Class Reference

This class defines four categories of operations upon int buffers:

```
#include <src/main/decaf/nio/IntBuffer.h>
```

Inheritance diagram for `decaf::nio::IntBuffer`:

### Public Member Functions

- `virtual ~IntBuffer ()`
- `virtual std::string toString () const`
- `virtual int * array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the int array that backs this buffer (optional operation).*
- `virtual int arrayOffset ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- `virtual IntBuffer * asReadOnlyBuffer () const =0`  
*Creates a new, read-only int buffer that shares this buffer's content.*
- `virtual IntBuffer & compact ()=0 throw ( ReadOnlyBufferException )`  
*Compacts this buffer.*

- virtual **IntBuffer** \* **duplicate** ()=0  
*Creates a new int buffer that shares this buffer's content.*
- virtual int **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual int **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **IntBuffer** & **get** (std::vector< int > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **IntBuffer** & **get** (int \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible int array.*
- **IntBuffer** & **put** (**IntBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the ints remaining in the given source buffer into this buffer.*
- **IntBuffer** & **put** (const int \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers ints into this buffer from the given source array.*
- **IntBuffer** & **put** (std::vector< int > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source ints array into this buffer.*
- virtual **IntBuffer** & **put** (int value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given integer into this buffer at the current position, and then increments the position.*
- virtual **IntBuffer** & **put** (int index, int value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given ints into this buffer at the given index.*
- virtual **IntBuffer** \* **slice** () const =0  
*Creates a new **IntBuffer** (p. 2026) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **IntBuffer** &value) const
- virtual bool **equals** (const **IntBuffer** &value) const
- virtual bool **operator==** (const **IntBuffer** &value) const
- virtual bool **operator<** (const **IntBuffer** &value) const

## Static Public Member Functions

- static **IntBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new Double buffer.*
- static **IntBuffer** \* **wrap** (int \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **IntBuffer** (p. 2026).*
- static **IntBuffer** \* **wrap** (std::vector< int > &buffer)  
*Wraps the passed STL int Vector in a **IntBuffer** (p. 2026).*

## Protected Member Functions

- **IntBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **IntBuffer** (p. 2026) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.404.1 Detailed Description

This class defines four categories of operations upon int buffers:

o Absolute and relative get and put methods that read and write single ints;  
 o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and  
 o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer  
 o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.404.2 Constructor & Destructor Documentation

6.404.2.1 decaf::nio::IntBuffer::IntBuffer ( int *capacity* ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **IntBuffer** (p. 2026) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 887) in integers.
-----------------	---

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if capacity is negative.
---------------------------------------	--------------------------

6.404.2.2 `virtual decaf::nio::IntBuffer::~IntBuffer ( ) [inline, virtual]`

**6.404.3 Member Function Documentation**

6.404.3.1 `static IntBuffer* decaf::nio::IntBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

**Parameters**

<i>capacity</i>	The size of the Double buffer in integers.
-----------------	--

**Returns**

the **IntBuffer** (p. 2026) that was allocated, caller owns.

6.404.3.2 `virtual int* decaf::nio::IntBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 887).

**Exceptions**

<b><i>ReadOnlyBufferEx- ception</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOpera- tionException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2021).

6.404.3.3 `virtual int decaf::nio::IntBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2021).

6.404.3.4 `virtual IntBuffer* decaf::nio::IntBuffer::asReadOnlyBuffer ( ) const [pure virtual]`

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only int buffer which the caller then owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2022).

6.404.3.5 `virtual IntBuffer& decaf::nio::IntBuffer::compact ( ) throw ( ReadOnlyBufferException ) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **IntBuffer** (p. 2026)

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2022).

6.404.3.6 `virtual int decaf::nio::IntBuffer::compareTo ( const IntBuffer & value ) const`  
[virtual]

6.404.3.7 `virtual IntBuffer* decaf::nio::IntBuffer::duplicate ( )` [pure virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new int **Buffer** (p. 887) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2023).

6.404.3.8 `virtual bool decaf::nio::IntBuffer::equals ( const IntBuffer & value ) const`  
[virtual]

6.404.3.9 `IntBuffer& decaf::nio::IntBuffer::get ( std::vector< int > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this **Buffer** (p. 887).

### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there are fewer than length ints remaining in this buffer.
---	---

6.404.3.10 `virtual int decaf::nio::IntBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the int is to be read.
--------------	--

### Returns

the int that is located at the given index.

### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2023).

6.404.3.11 `IntBuffer& decaf::nio::IntBuffer::get ( int * buffer, int size, int offset, int length ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Relative bulk get method.

This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 892), then no bytes are transferred and a **BufferUnderflowException** (p. 916) is thrown.

Otherwise, this method copies length ints from this buffer into the given array, starting

at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer that was passed in.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 887).

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length ints remaining in this buffer.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.404.3.12 `virtual int decaf::nio::IntBuffer::get ( ) throw ( BufferUnderflowException )`  
[pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the int at the current position.

#### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2023).

6.404.3.13 `virtual bool decaf::nio::IntBuffer::hasArray ( ) const` [pure virtual]

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.



Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2024).

6.404.3.14 virtual bool decaf::nio::IntBuffer::operator< ( const IntBuffer & *value* ) const  
[virtual]

6.404.3.15 virtual bool decaf::nio::IntBuffer::operator==( const IntBuffer & *value* ) const  
[virtual]

6.404.3.16 virtual IntBuffer& decaf::nio::IntBuffer::put ( int *value* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException** ) [pure  
virtual]

Writes the given integer into this buffer at the current position, and then increments the position.

Parameters

<i>value</i>	The integer value to be written.
--------------	----------------------------------

Returns

a reference to this buffer.

Exceptions

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2025).

6.404.3.17 virtual IntBuffer& decaf::nio::IntBuffer::put ( int *index*, int *value* )  
throw ( **decaf::lang::exceptions::IndexOutOfBoundsException**,  
**ReadOnlyBufferException** ) [pure virtual]

Writes the given ints into this buffer at the given index.

Parameters

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The ints to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	- If index greater than the buffer's limit minus the size of the type being written, or the index is negative.
<b>ReadOnlyBufferException</b> (p. 3115)	- If this buffer is read-only.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2024).

6.404.3.18 **IntArrayBuffer**& decaf::nio::IntArrayBuffer::put ( const int \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException**, decaf::lang::exceptions::**IndexOutOfBoundsException**, decaf::lang::exceptions::**NullPointerException** )

This method transfers ints into this buffer from the given source array.

If there are more ints to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 892), then no ints are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

**Parameters**

<i>buffer</i>	The array from which integers are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of integers to be read from the given array.

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

### 6.404.3.19 IntBuffer& decaf::nio::IntBuffer::put ( std::vector< int > & *buffer* ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source ints array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>IntBuffer</b> (p. 2026).
---------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

### 6.404.3.20 IntBuffer& decaf::nio::IntBuffer::put ( IntBuffer & *src* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )

This method transfers the ints remaining in the given source buffer into this buffer.

If there are more ints remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 892), then no ints are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `n = src.remaining()` ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take ints from an place in this one.
------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer for the remaining ints in the source buffer.
<b>IllegalArgumentException</b>	if the source buffer is this buffer.

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.
--	------------------------------

6.404.3.21 `virtual IntBuffer* decaf::nio::IntBuffer::slice ( ) const` [pure virtual]

Creates a new **IntBuffer** (p. 2026) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **IntBuffer** (p. 2026) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 2026).

6.404.3.22 `virtual std::string decaf::nio::IntBuffer::toString ( ) const` [virtual]

#### Returns

a `std::string` describing this object

6.404.3.23 `static IntBuffer* decaf::nio::IntBuffer::wrap ( int * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **IntBuffer** (p. 2026).

The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

Returns

a new **IntBuffer** (p. 2026) that is backed by buffer, caller owns.

Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.404.3.24    **static IntBuffer\* decaf::nio::IntBuffer::wrap ( std::vector< int > & buffer )**  
                  [static]

Wraps the passed STL int Vector in a **IntBuffer** (p. 2026).

The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

Returns

a new **IntBuffer** (p. 2026) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**IntBuffer.h**

6.405    decaf::lang::Integer Class Reference

#include <src/main/decaf/lang/Integer.h>

Inheritance diagram for decaf::lang::Integer:

Public Member Functions

- **Integer** (int value)
- **Integer** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Integer** ()
- virtual int **compareTo** (const **Integer** &i) const  
                  *Compares this **Integer** (p. 2038) instance with another.*

- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Integer** &i) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const int &i) const  
*Compares this **Integer** (p. 2038) instance with another.*
- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const int &i) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const
- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static **Integer decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3610) into a **Integer** (p. 2038).*
- static int **reverseBytes** (int value)  
*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.*
- static int **reverse** (int value)  
*Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.*
- static int **parseInt** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed int in the radix specified by the second argument.*
- static int **parseInt** (const std::string &s) throw ( exceptions::NumberFormatException )

*Parses the string argument as a signed decimal int.*

- static **Integer valueOf** (int value)

*Returns a **Integer** (p. 2038) instance representing the specified int value.*

- static **Integer valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )

*Returns a **Integer** (p. 2038) object holding the value given by the specified std::string.*

- static **Integer valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )

*Returns a **Integer** (p. 2038) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

- static int **bitCount** (int value)

*Returns the number of one-bits in the two's complement binary representation of the specified int value.*

- static std::string **toString** (int value)

*Converts the int to a **String** (p. 3610) representation.*

- static std::string **toString** (int value, int radix)

*Returns a string representation of the first argument in the radix specified by the second argument.*

- static std::string **toHexString** (int value)

*Returns a string representation of the integer argument as an unsigned integer in base 16.*

- static std::string **toOctalString** (int value)

*Returns a string representation of the integer argument as an unsigned integer in base 8.*

- static std::string **toBinaryString** (int value)

*Returns a string representation of the integer argument as an unsigned integer in base 2.*

- static int **highestOneBit** (int value)

*Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.*

- static int **lowestOneBit** (int value)

*Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.*

- static int **numberOfLeadingZeros** (int value)

*Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.*

- static int **numberOfTrailingZeros** (int value)

*Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.*

- static int **rotateLeft** (int value, int distance)

*Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.*

- static int **rotateRight** (int value, int distance)

*Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.*

- static int **signum** (int value)

*Returns the signum function of the specified int value.*

## Static Public Attributes

- static const int **SIZE** = 32  
*The size in bits of the primitive int type.*
- static const int **MAX\_VALUE** = (int)0x7FFFFFFF  
*The maximum value that the primitive type can hold.*
- static const int **MIN\_VALUE** = (int)0x80000000  
*The minimum value that the primitive type can hold.*

## 6.405.1 Constructor & Destructor Documentation

### 6.405.1.1 decaf::lang::Integer::Integer ( int *value* )

#### Parameters

<i>value</i>	The primitive value to wrap in an <b>Integer</b> (p.2038) instance.
--------------	---

### 6.405.1.2 decaf::lang::Integer::Integer ( const std::string & *value* ) throw ( exceptions::NumberFormatException )

#### Parameters

<i>value</i>	The base 10 encoded string to decode to an <b>Integer</b> (p.2038) and wrap.
--------------	--

#### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

### 6.405.1.3 virtual decaf::lang::Integer::~~Integer ( ) [inline, virtual]

## 6.405.2 Member Function Documentation

### 6.405.2.1 static int decaf::lang::Integer::bitCount ( int *value* ) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

#### Parameters

<i>value</i>	- the int to count
--------------	--------------------

#### Returns

the number of one-bits in the two's complement binary representation of the speci-



fied int value.

**6.405.2.2** `virtual unsigned char decaf::lang::Integer::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2787).

**6.405.2.3** `virtual int decaf::lang::Integer::compareTo ( const int & i ) const [virtual]`

Compares this **Integer** (p. 2038) instance with another.

#### Parameters

<i>i</i>	- the <b>Integer</b> (p. 2038) instance to be compared
----------	--

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< int > (p. 1187).

**6.405.2.4** `virtual int decaf::lang::Integer::compareTo ( const Integer & i ) const [virtual]`

Compares this **Integer** (p. 2038) instance with another.

#### Parameters

<i>i</i>	- the <b>Integer</b> (p. 2038) instance to be compared
----------	--

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< Integer > (p. 1187).

6.405.2.5 `static Integer decaf::lang::Integer::decode ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Decodes a **String** (p. 3610) into a **Integer** (p. 2038).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 3610) is the minus sign. No whitespace characters are permitted in the string.

#### Parameters

<i>value</i>	- The string to decode
--------------	------------------------

#### Returns

a **Integer** (p. 2038) object containing the decoded value

#### Exceptions

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

6.405.2.6 `virtual double decaf::lang::Integer::doubleValue ( ) const [inline, virtual]`

Answers the double value which the receiver represents.

#### Returns

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.405.2.7 `bool decaf::lang::Integer::equals ( const Integer & i ) const [inline, virtual]`

#### Parameters

<i>i</i>	- the <b>Integer</b> (p. 2038) object to compare against.
----------	---

#### Returns

true if the two **Integer** (p. 2038) Objects have the same value.

Implements **decaf::lang::Comparable**< **Integer** > (p. 1188).

6.405.2.8 `bool decaf::lang::Integer::equals ( const int & i ) const [inline, virtual]`

#### Parameters

<i>i</i>	- the <b>Integer</b> (p. 2038) object to compare against.
----------	---

#### Returns

true if the two **Integer** (p. 2038) Objects have the same value.

Implements **decaf::lang::Comparable**< **int** > (p. 1188).

6.405.2.9 `virtual float decaf::lang::Integer::floatValue ( ) const [inline, virtual]`

Answers the float value which the receiver represents.

#### Returns

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.405.2.10 `static int decaf::lang::Integer::highestOneBit ( int value ) [static]`

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

#### Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

#### Returns

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.405.2.11 `virtual int decaf::lang::Integer::intValue ( ) const [inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.405.2.12 `virtual long long decaf::lang::Integer::longValue ( ) const [inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.405.2.13 `static int decaf::lang::Integer::lowestOneBit ( int value ) [static]`

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

#### Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

#### Returns

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.405.2.14 `static int decaf::lang::Integer::numberOfLeadingZeros ( int value ) [static]`

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values  $x$ :

$\ast \text{ floor}(\log_2(x)) = 31 - \text{numberOfLeadingZeros}(x) \ast \text{ ceil}(\log_2(x)) = 32 - \text{numberOfLeadingZeros}(x - 1)$

#### Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

#### Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.405.2.15 `static int decaf::lang::Integer::numberOfTrailingZeros ( int value ) [static]`

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

#### Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

#### Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.405.2.16 `virtual bool decaf::lang::Integer::operator< ( const int & i ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>i</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< int >` (p. 1188).

6.405.2.17 `virtual bool decaf::lang::Integer::operator< ( const Integer & i ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>i</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< Integer >` (p. 1188).

6.405.2.18 `virtual bool decaf::lang::Integer::operator==( const Integer & i ) const`  
`[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<i>i</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< Integer > (p. 1189).

6.405.2.19 `virtual bool decaf::lang::Integer::operator==( const int & i ) const` `[inline, virtual]`

Compares equality between this object and the one passed.

#### Parameters

<i>i</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< int > (p. 1189).

6.405.2.20 `static int decaf::lang::Integer::parseInt ( const std::string & s, int radix ) throw (`  
`exceptions::NumberFormatException )` `[static]`

Parses the string argument as a signed int in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1072) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs: \* The first argument is null or is a string of length zero. \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1076) or larger than **Character.MAX\_RADIX** (p. 1076). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type int.

#### Parameters

<i>s</i>	- the <b>String</b> (p. 3610) containing the int representation to be parsed
<i>radix</i>	- the radix to be used while parsing s

**Returns**

the int represented by the string argument in the specified radix.

**Exceptions**

<i>NumberFormatException</i>	- If <b>String</b> (p. 3610) does not contain a parsable int.
------------------------------	---

**6.405.2.21** static int decaf::lang::Integer::parseInt ( const std::string & s ) throw ( exceptions::NumberFormatException ) [static]

Parses the string argument as a signed decimal int.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseInt( const std::string, int ) method.

**Parameters**

s	- <b>String</b> (p. 3610) to convert to a int
---	---

**Returns**

the converted int value

**Exceptions**

<i>NumberFormatException</i>	if the string is not a int.
------------------------------	-----------------------------

**6.405.2.22** static int decaf::lang::Integer::reverse ( int value ) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

**Parameters**

value	- the value whose bits are to be reversed
-------	---

**Returns**

the reversed bits int.

**6.405.2.23** static int decaf::lang::Integer::reverseBytes ( int value ) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

**Parameters**

<i>value</i>	- the int whose bytes we are to reverse
--------------	---

**Returns**

the reversed int.

**6.405.2.24** `static int decaf::lang::Integer::rotateLeft ( int value, int distance )` `[static]`

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: `rotateLeft(val, -distance) == rotateRight(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F)`.

**Parameters**

<i>value</i>	- the int to be inspected
<i>distance</i>	- the number of bits to rotate

**Returns**

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

**6.405.2.25** `static int decaf::lang::Integer::rotateRight ( int value, int distance )` `[static]`

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

**Parameters**

<i>value</i>	- the int to be inspected
<i>distance</i>	- the number of bits to rotate

**Returns**

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.



6.405.2.26 `virtual short decaf::lang::Integer::shortValue ( ) const [inline, virtual]`

Answers the short value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p.2788).

6.405.2.27 `static int decaf::lang::Integer::signum ( int value ) [static]`

Returns the signum function of the specified int value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

#### Parameters

<i>value</i>	- the int to be inspected
--------------	---------------------------

#### Returns

the signum function of the specified int value.

6.405.2.28 `static std::string decaf::lang::Integer::toBinaryString ( int value ) [static]`

Returns a string representation of the integer argument as an unsigned integer in base 2.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

#### Parameters

<i>value</i>	- the int to be translated to a binary string
--------------	---

#### Returns

the unsigned int value as a binary string

6.405.2.29 `static std::string decaf::lang::Integer::toHexString ( int value ) [static]`

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the `toUpperCase()` method may be called on the result:

#### Parameters

<i>value</i>	- the int to be translated to an Octal string
--------------	---

#### Returns

the unsigned int value as a Octal string

6.405.2.30 `static std::string decaf::lang::Integer::toOctalString ( int value ) [static]`

Returns a string representation of the integer argument as an unsigned integer in base 8.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

#### Parameters

<i>value</i>	- the int to be translated to an Octal string
--------------	---

#### Returns

the unsigned int value as a Octal string

6.405.2.31 `std::string decaf::lang::Integer::toString ( ) const`

#### Returns

this **Integer** (p. 2038) Object as a **String** (p. 3610) Representation

6.405.2.32 `static std::string decaf::lang::Integer::toString ( int value, int radix )` `[static]`

Returns a string representation of the first argument in the radix specified by the second argument.

If the radix is smaller than **Character.MIN\_RADIX** (p. 1076) or larger than **Character.MAX\_RADIX** (p. 1076), then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

#### Parameters

<i>value</i>	- the int to convert to a string
<i>radix</i>	- the radix to format the string in

#### Returns

an int formatted to the string value of the radix given.

6.405.2.33 `static std::string decaf::lang::Integer::toString ( int value )` `[static]`

Converts the int to a **String** (p. 3610) representation.

#### Parameters

<i>value</i>	The int to convert to a <code>std::string</code> instance.
--------------	--

#### Returns

string representation

6.405.2.34 `static Integer decaf::lang::Integer::valueOf ( const std::string & value, int radix )`  
`throw ( exceptions::NumberFormatException )` `[static]`

Returns a **Integer** (p. 2038) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the `parseInt( std::string, int )` method. The result is a **Integer** (p. 2038) object that represents the int value specified by the string.

#### Parameters

<i>value</i>	- std::string to parse as base ( radix )
<i>radix</i>	- base of the string to parse.

**Returns**

new **Integer** (p. 2038) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a valid int.
------------------------------	-----------------------------------

6.405.2.35 **static Integer decaf::lang::Integer::valueOf ( int value )** [inline, static]

Returns a **Integer** (p. 2038) instance representing the specified int value.

**Parameters**

<i>value</i>	- the int to wrap
--------------	-------------------

**Returns**

the new **Integer** (p. 2038) object wrapping value.

6.405.2.36 **static Integer decaf::lang::Integer::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException )** [static]

Returns a **Integer** (p. 2038) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the parseInt( std::string ) method. The result is a **Integer** (p. 2038) object that represents the int value specified by the string.

**Parameters**

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

**Returns**

new **Integer** (p. 2038) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a decimal int.
------------------------------	-------------------------------------

### 6.405.3 Field Documentation

6.405.3.1 `const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF` `[static]`

The maximum value that the primitive type can hold.

6.405.3.2 `const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000` `[static]`

The minimum value that the primitive type can hold.

6.405.3.3 `const int decaf::lang::Integer::SIZE = 32` `[static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Integer.h`

## 6.406 activemq::commands::IntegerResponse Class Reference

```
#include <src/main/activemq/commands/IntegerResponse.h>
```

Inheritance diagram for `activemq::commands::IntegerResponse`:

### Public Member Functions

- **IntegerResponse** ()
- virtual `~IntegerResponse` ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **IntegerResponse** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual int **getResult** () const
- virtual void **setResult** (int result)

### Static Public Attributes

- static const unsigned char **ID\_INTEGERRESPONSE** = 34

### Protected Attributes

- int **result**

## 6.406.1 Constructor & Destructor Documentation

6.406.1.1 `activemq::commands::IntegerResponse::IntegerResponse ( )`

6.406.1.2 `virtual activemq::commands::IntegerResponse::~~IntegerResponse ( )`  
[virtual]

## 6.406.2 Member Function Documentation

6.406.2.1 `virtual IntegerResponse* activemq::commands::IntegerResponse::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 3228).

6.406.2.2 `virtual void activemq::commands::IntegerResponse::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::Response** (p. 3229).

6.406.2.3 `virtual bool activemq::commands::IntegerResponse::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 3229).

6.406.2.4 virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType ( ) const [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::Response** (p. 3230).

6.406.2.5 virtual int activemq::commands::IntegerResponse::getResult ( ) const [virtual]

6.406.2.6 virtual void activemq::commands::IntegerResponse::setResult ( int result ) [virtual]

6.406.2.7 virtual std::string activemq::commands::IntegerResponse::toString ( ) const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 3230).

## 6.406.3 Field Documentation

6.406.3.1 const unsigned char activemq::commands::IntegerResponse::ID\_ - INTEGERRESPONSE = 34 [static]

6.406.3.2 int activemq::commands::IntegerResponse::result [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/IntegerResponse.h

## 6.407 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2057).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.407.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2057).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.407.2 Constructor & Destructor Documentation

6.407.2.1 `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]`

6.407.2.2 `virtual activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::~~IntegerResponseMarshaller ( ) [inline, virtual]`

## 6.407.3 Member Function Documentation

6.407.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3261).

6.407.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v6::ResponseMarshaller` (p. 3261).

6.407.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3261).

```
6.407.3.4  virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262).

```
6.407.3.5  virtual int activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.407 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller

### Class Reference 2069

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3262).

```
6.407.3.6 virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3263).

```
6.407.3.7 virtual void activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** (p. 3264).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**IntegerResponseMarshaller.h**

## 6.408 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2061).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.408.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2061).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.408.2 Constructor & Destructor Documentation

6.408.2.1 `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]`

6.408.2.2 `virtual activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::~~IntegerResponseMarshaller ( ) [inline, virtual]`

## 6.408.3 Member Function Documentation

6.408.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3242).

6.408.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3242).

6.408.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.408.3.4  virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3243).

```
6.408.3.5  virtual int activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.408 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller

### Class Reference 2073

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

```
6.408.3.6 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3244).

```
6.408.3.7 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 3245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h

## 6.409 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2065).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.409.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2065).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.409.2 Constructor & Destructor Documentation

6.409.2.1 `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]`

6.409.2.2 `virtual activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::~~IntegerResponseMarshaller ( ) [inline, virtual]`

## 6.409.3 Member Function Documentation

6.409.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3251).

6.409.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

6.409.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3252).

```
6.409.3.4  virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253).

```
6.409.3.5  virtual int activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.409 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller

### Class Reference 2077

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3253).

```
6.409.3.6 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254).

```
6.409.3.7 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 3254).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**IntegerResponseMarshaller.h**

## 6.410 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2069).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.410.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2069).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.410.2 Constructor & Destructor Documentation

6.410.2.1 `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]`

6.410.2.2 `virtual activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::~~IntegerResponseMarshaller ( ) [inline, virtual]`

## 6.410.3 Member Function Documentation

6.410.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3237).

6.410.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

6.410.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3238).

```
6.410.3.4  virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

```
6.410.3.5  virtual int activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.410 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller

### Class Reference 2081

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3239).

6.410.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

6.410.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 3240).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**IntegerResponseMarshaller.h**

## 6.411 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2073).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.411.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2073).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.411.2 Constructor & Destructor Documentation

6.411.2.1 `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]`

6.411.2.2 `virtual activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::~~IntegerResponseMarshaller ( ) [inline, virtual]`

## 6.411.3 Member Function Documentation

6.411.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

6.411.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3256).

6.411.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.411.3.4  virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3257).

```
6.411.3.5  virtual int activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.411 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller

### Class Reference 2085

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

6.411.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3258).

6.411.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 3259).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**IntegerResponseMarshaller.h**

## 6.412 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2077).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller:

### Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.412.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2077).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.412.2 Constructor & Destructor Documentation

6.412.2.1 `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::IntegerResponseMarshaller ( ) [inline]`

6.412.2.2 `virtual activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::~~IntegerResponseMarshaller ( ) [inline, virtual]`

## 6.412.3 Member Function Documentation

6.412.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

6.412.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

6.412.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3247).

6.412.3.4 **virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseUnmarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )**  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

6.412.3.5 **virtual int activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal1**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,**  
**utils::BooleanStream \* bs ) throw ( decaf::io::IOException )**  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.412 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller

### Class Reference 2089

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3248).

6.412.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3249).

6.412.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 3250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**IntegerResponseMarshaller.h**

## 6.413 internal\_state Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

### Data Fields

- **z\_stream** strm
- **int** status
- **Bytef** \* pending\_buf
- **ulg** pending\_buf\_size
- **Bytef** \* pending\_out
- **ulnt** pending
- **int** wrap
- **gz\_headerp** gzhead
- **ulnt** gzindex
- **Byte** method
- **int** last\_flush
- **ulnt** w\_size
- **ulnt** w\_bits
- **ulnt** w\_mask
- **Bytef** \* window
- **ulg** window\_size
- **Posf** \* prev
- **Posf** \* head
- **ulnt** ins\_h
- **ulnt** hash\_size
- **ulnt** hash\_bits
- **ulnt** hash\_mask
- **ulnt** hash\_shift
- **long** block\_start
- **ulnt** match\_length
- **IPos** prev\_match
- **int** match\_available
- **ulnt** strstart
- **ulnt** match\_start
- **ulnt** lookahead
- **ulnt** prev\_length
- **ulnt** max\_chain\_length
- **ulnt** max\_lazy\_match
- **int** level
- **int** strategy
- **ulnt** good\_match
- **int** nice\_match
- **struct** **ct\_data\_s** dyn\_ltree [HEAP\_SIZE]
- **struct** **ct\_data\_s** dyn\_dtree [2 \*D\_CODES+1]
- **struct** **ct\_data\_s** bl\_tree [2 \*BL\_CODES+1]



- struct **tree\_desc\_s** **l\_desc**
- struct **tree\_desc\_s** **d\_desc**
- struct **tree\_desc\_s** **bl\_desc**
- **ush** **bl\_count** [MAX\_BITS+1]
- **int** **heap** [2 \*L\_CODES+1]
- **int** **heap\_len**
- **int** **heap\_max**
- **uch** **depth** [2 \*L\_CODES+1]
- **uchf** \* **l\_buf**
- **uint** **lit\_bufsize**
- **uint** **last\_lit**
- **ushf** \* **d\_buf**
- **ulg** **opt\_len**
- **ulg** **static\_len**
- **uint** **matches**
- **int** **last\_eob\_len**
- **ush** **bi\_buf**
- **int** **bi\_valid**
- **ulg** **high\_water**
- **int** **dummy**

#### 6.413.1 Field Documentation

6.413.1.1 **ush** **internal\_state::bi\_buf**

6.413.1.2 **int** **internal\_state::bi\_valid**

6.413.1.3 **ush** **internal\_state::bl\_count**[MAX\_BITS+1]

6.413.1.4 struct **tree\_desc\_s** **internal\_state::bl\_desc**

6.413.1.5 struct **ct\_data\_s** **internal\_state::bl\_tree**[2 \*BL\_CODES+1]

6.413.1.6 **long** **internal\_state::block\_start**

6.413.1.7 **ushf**\* **internal\_state::d\_buf**

6.413.1.8 struct **tree\_desc\_s** **internal\_state::d\_desc**

6.413.1.9 **uch** **internal\_state::depth**[2 \*L\_CODES+1]

6.413.1.10 **int** **internal\_state::dummy**

6.413.1.11 struct **ct\_data\_s** **internal\_state::dyn\_dtree**[2 \*D\_CODES+1]

6.413.1.12 struct **ct\_data\_s** **internal\_state::dyn\_ltree**[HEAP\_SIZE]

- 6.413.1.13 `ulnt internal_state::good_match`
- 6.413.1.14 `gz_headerp internal_state::gzhead`
- 6.413.1.15 `ulnt internal_state::gzindex`
- 6.413.1.16 `ulnt internal_state::hash_bits`
- 6.413.1.17 `ulnt internal_state::hash_mask`
- 6.413.1.18 `ulnt internal_state::hash_shift`
- 6.413.1.19 `ulnt internal_state::hash_size`
- 6.413.1.20 `Posf* internal_state::head`
- 6.413.1.21 `int internal_state::heap[2 * L_CODES+1]`
- 6.413.1.22 `int internal_state::heap_len`
- 6.413.1.23 `int internal_state::heap_max`
- 6.413.1.24 `ulg internal_state::high_water`
- 6.413.1.25 `ulnt internal_state::ins_h`
- 6.413.1.26 `uchf* internal_state::l_buf`
- 6.413.1.27 `struct tree_desc_s internal_state::l_desc`
- 6.413.1.28 `int internal_state::last_eob_len`
- 6.413.1.29 `int internal_state::last_flush`
- 6.413.1.30 `ulnt internal_state::last_lit`
- 6.413.1.31 `int internal_state::level`
- 6.413.1.32 `ulnt internal_state::lit_bufsize`
- 6.413.1.33 `ulnt internal_state::lookahead`
- 6.413.1.34 `int internal_state::match_available`
- 6.413.1.35 `ulnt internal_state::match_length`
- 6.413.1.36 `ulnt internal_state::match_start`

- 6.413.1.37 `uint internal_state::matches`
- 6.413.1.38 `uint internal_state::max_chain_length`
- 6.413.1.39 `uint internal_state::max_lazy_match`
- 6.413.1.40 `Byte internal_state::method`
- 6.413.1.41 `int internal_state::nice_match`
- 6.413.1.42 `ulg internal_state::opt_len`
- 6.413.1.43 `uint internal_state::pending`
- 6.413.1.44 `Bytef* internal_state::pending_buf`
- 6.413.1.45 `ulg internal_state::pending_buf_size`
- 6.413.1.46 `Bytef* internal_state::pending_out`
- 6.413.1.47 `Posf* internal_state::prev`
- 6.413.1.48 `uint internal_state::prev_length`
- 6.413.1.49 `IPos internal_state::prev_match`
- 6.413.1.50 `ulg internal_state::static_len`
- 6.413.1.51 `int internal_state::status`
- 6.413.1.52 `int internal_state::strategy`
- 6.413.1.53 `z_streamp internal_state::strm`
- 6.413.1.54 `uint internal_state::strstart`
- 6.413.1.55 `uint internal_state::w_bits`
- 6.413.1.56 `uint internal_state::w_mask`
- 6.413.1.57 `uint internal_state::w_size`
- 6.413.1.58 `Bytef* internal_state::window`
- 6.413.1.59 `ulg internal_state::window_size`

## 6.413.1.60 int internal\_state::wrap

The documentation for this struct was generated from the following files:

- src/main/decaf/internal/util/zip/**deflate.h**
- src/main/decaf/internal/util/zip/**zlib.h**

## 6.414 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 2724).

```
#include <src/main/activemq/transport/mock/InternalCommandListener.h>
```

Inheritance diagram for activemq::transport::mock::InternalCommandListener:

### Public Member Functions

- **InternalCommandListener** ()
- virtual **~InternalCommandListener** ()
- void **setTransport** (**MockTransport** \*transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &response-Builder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- void **run** ()  
*Default implementation of the run method - does nothing.*

#### 6.414.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 2724).

This class processes all outbound commands and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 3231) and getting a set of Commands to send back into the **MockTransport** (p. 2724) as incoming Commands and Responses.

#### 6.414.2 Constructor & Destructor Documentation

6.414.2.1 **activemq::transport::mock::InternalCommandListener::InternalCommandListener** ( )

6.414.2.2 **virtual activemq::transport::mock::InternalCommandListener::~~InternalCommandListener** ( ) [virtual]

### 6.414.3 Member Function Documentation

6.414.3.1 `virtual void activemq::transport::mock::InternalCommandListener::onCommand ( const Pointer< Command > & command ) [virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3819) deletes the command upon receipt.

#### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implements **activemq::transport::TransportListener** (p. 3836).

6.414.3.2 `void activemq::transport::mock::InternalCommandListener::run ( ) [virtual]`

Default implementation of the run method - does nothing.

Reimplemented from **decaf::lang::Thread** (p. 3713).

6.414.3.3 `void activemq::transport::mock::InternalCommandListener::setResponseBuilder ( const Pointer< ResponseBuilder > & responseBuilder ) [inline]`

6.414.3.4 `void activemq::transport::mock::InternalCommandListener::setTransport ( MockTransport * transport ) [inline]`

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**InternalCommandListener.h**

## 6.415 decaf::lang::exceptions::InterruptedException Class Reference

```
#include <src/main/decaf/lang/exceptions/InterruptedException.h>
```

Inheritance diagram for decaf::lang::exceptions::InterruptedException:

### Public Member Functions

- **InterruptedException** () throw ()  
*Default Constructor.*
- **InterruptedException** (const **Exception** &ex) throw ()

*Conversion Constructor from some other **Exception** (p. 1794).*

- **InterruptedException** (const **InterruptedException** &ex) throw ()

*Copy Constructor.*

- **InterruptedException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **InterruptedException** (const std::exception \*cause) throw ()

*Constructor.*

- **InterruptedException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **InterruptedException** \* clone () const

*Clones this exception.*

- virtual ~**InterruptedException** () throw ()

## 6.415.1 Constructor & Destructor Documentation

6.415.1.1 **decaf::lang::exceptions::InterruptedException::InterruptedException ( ) throw ()**  
[inline]

Default Constructor.

6.415.1.2 **decaf::lang::exceptions::InterruptedException::InterruptedException ( const Exception & ex ) throw ()** [inline]

Conversion Constructor from some other **Exception** (p. 1794).

### Parameters

ex	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
----	--

6.415.1.3 **decaf::lang::exceptions::InterruptedException::InterruptedException ( const InterruptedException & ex ) throw ()** [inline]

Copy Constructor.

### Parameters

ex	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
----	--

6.415.1.4 `decaf::lang::exceptions::InterruptedException::InterruptedException ( const char *  
file, const int lineNumber, const std::exception * cause, const char * msg, ... )  
throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.415.1.5 `decaf::lang::exceptions::InterruptedException::InterruptedException ( const  
std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p.2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.415.1.6 `decaf::lang::exceptions::InterruptedException::InterruptedException ( const char *  
file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.415.1.7 `virtual decaf::lang::exceptions::InterruptedException::~InterruptedException ( )  
throw () [inline, virtual]`

## 6.415.2 Member Function Documentation

```
6.415.2.1 virtual InterruptedException* de-
caf::lang::exceptions::InterruptedException::clone ( ) const
[inline, virtual]
```

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InterruptedException.h**

## 6.416 decaf::io::InterruptedIOException Class Reference

```
#include <src/main/decaf/io/InterruptedIOException.h>
```

Inheritance diagram for decaf::io::InterruptedIOException:

### Public Member Functions

- **InterruptedIOException** () throw ()  
*Default Constructor.*
- **InterruptedIOException** (const **lang::Exception** &ex) throw ()  
*Copy Constructor.*
- **InterruptedIOException** (const **InterruptedIOException** &ex) throw ()  
*Copy Constructor.*
- **InterruptedIOException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InterruptedIOException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InterruptedIOException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **InterruptedIOException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InterruptedIOException** () throw ()



### 6.416.1 Constructor & Destructor Documentation

6.416.1.1 `decaf::io::InterruptedIOException::InterruptedIOException ( ) throw () [inline]`

Default Constructor.

6.416.1.2 `decaf::io::InterruptedIOException::InterruptedIOException ( const lang::Exception & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.416.1.3 `decaf::io::InterruptedIOException::InterruptedIOException ( const InterruptedIOException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.416.1.4 `decaf::io::InterruptedIOException::InterruptedIOException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.416.1.5 `decaf::io::InterruptedIOException::InterruptedIOException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.416.1.6 `decaf::io::InterruptedIOException::InterruptedIOException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.416.1.7 `virtual decaf::io::InterruptedIOException::~~InterruptedIOException ( ) throw () [inline, virtual]`

**6.416.2 Member Function Documentation**

6.416.2.1 `virtual InterruptedIOException* decaf::io::InterruptedIOException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2105).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 3489).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InterruptedIOException.h`

**6.417 cms::InvalidClientIdException Class Reference**

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

```
#include <src/main/cms/InvalidClientIdException.h>
```

Inheritance diagram for cms::InvalidClientIdException:

## Public Member Functions

- **InvalidClientIdException** () throw ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidClientIdException** () throw ()

### 6.417.1 Detailed Description

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

#### Since

1.3

### 6.417.2 Constructor & Destructor Documentation

6.417.2.1 cms::InvalidClientIdException::InvalidClientIdException ( ) throw ()

6.417.2.2 cms::InvalidClientIdException::InvalidClientIdException ( const InvalidClientIdException & ex ) throw ()

6.417.2.3 cms::InvalidClientIdException::InvalidClientIdException ( const std::string & message, const std::exception \* cause ) throw ()

6.417.2.4 cms::InvalidClientIdException::InvalidClientIdException ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.417.2.5 virtual cms::InvalidClientIdException::~~InvalidClientIdException ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidClientIdException.h**

## 6.418 cms::InvalidDestinationException Class Reference

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

```
#include <src/main/cms/InvalidDestinationException.h>
```

Inheritance diagram for cms::InvalidDestinationException:

### Public Member Functions

- **InvalidDestinationException** () throw ()
- **InvalidDestinationException** (const **InvalidDestinationException** &ex) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception \*cause) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidDestinationException** () throw ()

### 6.418.1 Detailed Description

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

#### Since

1.3

### 6.418.2 Constructor & Destructor Documentation

6.418.2.1 cms::InvalidDestinationException::InvalidDestinationException ( ) throw ()

6.418.2.2 cms::InvalidDestinationException::InvalidDestinationException ( const InvalidDestinationException &ex ) throw ()

6.418.2.3 cms::InvalidDestinationException::InvalidDestinationException ( const std::string &message, const std::exception \* cause ) throw ()

6.418.2.4 cms::InvalidDestinationException::InvalidDestinationException ( const std::string &message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > &stackTrace ) throw ()

6.418.2.5 virtual cms::InvalidDestinationException::~InvalidDestinationException ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidDestinationException.h**

## 6.419 decaf::security::InvalidKeyException Class Reference

```
#include <src/main/decaf/security/InvalidKeyException.h>
```

Inheritance diagram for decaf::security::InvalidKeyException:

### Public Member Functions

- **InvalidKeyException** () throw ()  
*Default Constructor.*
- **InvalidKeyException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()  
*Copy Constructor.*
- **InvalidKeyException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InvalidKeyException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **InvalidKeyException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **InvalidKeyException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InvalidKeyException** () throw ()

### 6.419.1 Constructor & Destructor Documentation

6.419.1.1 decaf::security::InvalidKeyException::InvalidKeyException ( ) throw () [inline]

Default Constructor.

6.419.1.2 `decaf::security::InvalidKeyException::InvalidKeyException ( const Exception & ex )  
throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.419.1.3 `decaf::security::InvalidKeyException::InvalidKeyException ( const  
InvalidKeyException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.419.1.4 `decaf::security::InvalidKeyException::InvalidKeyException ( const char * file, const  
int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()  
[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.419.1.5 `decaf::security::InvalidKeyException::InvalidKeyException ( const std::exception *  
cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.419.1.6 `decaf::security::InvalidKeyException::InvalidKeyException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.419.1.7 `virtual decaf::security::InvalidKeyException::~~InvalidKeyException ( ) throw ()`  
`[inline, virtual]`

## 6.419.2 Member Function Documentation

6.419.2.1 `virtual InvalidKeyException* decaf::security::InvalidKeyException::clone ( )`  
`const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 2257).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/InvalidKeyException.h`

## 6.420 decaf::nio::InvalidMarkException Class Reference

```
#include <src/main/decaf/nio/InvalidMarkException.h>
```

Inheritance diagram for decaf::nio::InvalidMarkException:

#### Public Member Functions

- **InvalidMarkException** () throw ()

*Default Constructor.*

- **InvalidMarkException** (const lang::Exception &ex) throw ()

*Conversion Constructor from some other Exception.*

- **InvalidMarkException** (const InvalidMarkException &ex) throw ()

*Copy Constructor.*

- **InvalidMarkException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **InvalidMarkException** (const std::exception \*cause) throw ()

*Constructor.*

- **InvalidMarkException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **InvalidMarkException** \* clone () const

*Clones this exception.*

- virtual ~**InvalidMarkException** () throw ()

## 6.420.1 Constructor & Destructor Documentation

6.420.1.1 **decaf::nio::InvalidMarkException::InvalidMarkException ( ) throw ()** [inline]

Default Constructor.

6.420.1.2 **decaf::nio::InvalidMarkException::InvalidMarkException ( const lang::Exception & ex ) throw ()** [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	The Exception whose state data is to be copied into this Exception.
-----------	---

6.420.1.3 **decaf::nio::InvalidMarkException::InvalidMarkException ( const InvalidMarkException & ex ) throw ()** [inline]

Copy Constructor.

### Parameters

<i>ex</i>	The Exception whose state data is to be copied into this Exception.
-----------	---



6.420.1.4 `decaf::nio::InvalidMarkException::InvalidMarkException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.420.1.5 `decaf::nio::InvalidMarkException::InvalidMarkException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.420.1.6 `decaf::nio::InvalidMarkException::InvalidMarkException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.420.1.7 `virtual decaf::nio::InvalidMarkException::~InvalidMarkException ( ) throw ()` `[inline, virtual]`

## 6.420.2 Member Function Documentation

6.420.2.1 `virtual InvalidMarkException* decaf::nio::InvalidMarkException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A new Exception instance that is a copy of this Exception.

Reimplemented from `decaf::lang::exceptions::IllegalStateException` (p. 1961).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/InvalidMarkException.h`

## 6.421 cms::InvalidSelectorException Class Reference

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

```
#include <src/main/cms/InvalidSelectorException.h>
```

Inheritance diagram for `cms::InvalidSelectorException`:

### Public Member Functions

- `InvalidSelectorException () throw ()`
- `InvalidSelectorException (const InvalidSelectorException &ex) throw ()`
- `InvalidSelectorException (const std::string &message, const std::exception *cause) throw ()`
- `InvalidSelectorException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~InvalidSelectorException () throw ()`

### 6.421.1 Detailed Description

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

### Since

1.3

### 6.421.2 Constructor & Destructor Documentation

6.421.2.1 `cms::InvalidSelectorException::InvalidSelectorException ( ) throw ()`

6.421.2.2 `cms::InvalidSelectorException::InvalidSelectorException ( const InvalidSelectorException & ex ) throw ()`

6.421.2.3 `cms::InvalidSelectorException::InvalidSelectorException ( const std::string & message, const std::exception * cause ) throw ()`

6.421.2.4 `cms::InvalidSelectorException::InvalidSelectorException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`

6.421.2.5 `virtual cms::InvalidSelectorException::~~InvalidSelectorException ( ) throw ()`  
[virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidSelectorException.h`

## 6.422 decaf::lang::exceptions::InvalidStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/InvalidStateException.h>
```

Inheritance diagram for `decaf::lang::exceptions::InvalidStateException`:

### Public Member Functions

- **InvalidStateException** ( ) throw ()  
*Default Constructor.*
- **InvalidStateException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **InvalidStateException** (const **InvalidStateException** &ex) throw ()  
*Copy Constructor.*
- **InvalidStateException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **InvalidStateException** (const std::exception \*cause) throw ()  
*Constructor.*
- **InvalidStateException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **InvalidStateException** \* **clone** () const  
*Clones this exception.*
- virtual ~**InvalidStateException** () throw ()

### 6.422.1 Constructor & Destructor Documentation

6.422.1.1 `decaf::lang::exceptions::InvalidStateException::InvalidStateException ( ) throw ()`  
[inline]

Default Constructor.

6.422.1.2 `decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const Exception & ex ) throw ()` [inline]

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.422.1.3 `decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const InvalidStateException & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.422.1.4 `decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.422.1.5 `decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.422.1.6 `decaf::lang::exceptions::InvalidStateException::InvalidStateException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.422.1.7 `virtual decaf::lang::exceptions::InvalidStateException::~~InvalidStateException ( ) throw () [inline, virtual]`

## 6.422.2 Member Function Documentation

6.422.2.1 `virtual InvalidStateException* decaf::lang::exceptions::InvalidStateException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InvalidStateException.h`

## 6.423 decaf::io::IOException Class Reference

```
#include <src/main/decaf/io/IOException.h>
```

Inheritance diagram for decaf::io::IOException:

### Public Member Functions

- **IOException** () throw ()  
*Default Constructor.*
- **IOException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **IOException** (const IOException &ex) throw ()  
*Copy Constructor.*
- **IOException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **IOException** (const std::exception \*cause) throw ()  
*Constructor.*
- **IOException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **IOException** \* clone () const  
*Clones this exception.*
- virtual ~**IOException** () throw ()

### 6.423.1 Constructor & Destructor Documentation

#### 6.423.1.1 decaf::io::IOException::IOException ( ) throw () [inline]

Default Constructor.

#### 6.423.1.2 decaf::io::IOException::IOException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.423.1.3 decaf::io::IOException::IOException ( const IOException & ex ) throw ()  
[inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.423.1.4 decaf::io::IOException::IOException ( const char \* file, const int lineNumber, const  
std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.423.1.5 decaf::io::IOException::IOException ( const std::exception \* cause ) throw ()  
[inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.423.1.6 decaf::io::IOException::IOException ( const char \* file, const int lineNumber, const  
char \* msg, ... ) throw () [inline]

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.423.1.7 `virtual decaf::io::IOException::~~IOException ( ) throw () [inline, virtual]`

## 6.423.2 Member Function Documentation

6.423.2.1 `virtual IOException* decaf::io::IOException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1797).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2824), **decaf::io::EOFException** (p. 1791), **decaf::io::InterruptedIOException** (p. 2091), **decaf::io::UnsupportedEncodingException** (p. 3849), **decaf::io::UTFDataFormatException** (p. 3900), **decaf::net::BindException** (p. 800), **decaf::net::ConnectException** (p. 1232), **decaf::net::HttpRetryException** (p. 1950), **decaf::net::MalformedURLException** (p. 2418), **decaf::net::NoRouteToHostException** (p. 2775), **decaf::net::PortUnreachableException** (p. 2924), **decaf::net::ProtocolException** (p. 3085), **decaf::net::SocketException** (p. 3467), **decaf::net::SocketTimeoutException** (p. 3489), **decaf::net::UnknownHostException** (p. 3844), **decaf::net::UnknownServiceException** (p. 3846), and **decaf::util::zip::ZipException** (p. 3993).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/IOException.h`

## 6.424 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 3819) interface that performs marshaling of commands to IO streams.

```
#include <src/main/activemq/transport/IOTransport.h>
```

Inheritance diagram for `activemq::transport::IOTransport`:

### Public Member Functions

- **IOTransport ( )**  
*Default Constructor.*



- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
 

*Create an instance of this **Transport** (p. 3819) and assign its **WireFormat** instance at creation time.*
- virtual ~**IOTransport** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )
 

*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )
 

*Not supported by this class - throws an exception.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )
 

*Not supported by this class - throws an exception.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
 

*Sets the **WireFormat** instance to use.*
- virtual void **setTransportListener** (**TransportListener** \*listener)
 

*Sets the observer of asynchronous exceptions from this transport.*
- virtual **TransportListener** \* **getTransportListener** () const
 

*Gets the observer of asynchronous exceptions from this transport.*
- virtual void **setInputStream** (decaf::io::DataInputStream \*is)
 

*Sets the input stream for in-coming commands.*
- virtual void **setOutputStream** (decaf::io::DataOutputStream \*os)
 

*Sets the output stream for out-going commands.*
- virtual void **start** () throw ( decaf::io::IOException )
 

*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **stop** () throw ( decaf::io::IOException )
 

*Stops the **Transport** (p. 3819), terminating any threads and stopping all read and write operations.*
- virtual void **close** () throw ( decaf::io::IOException )
 

*Stops the polling thread and closes the streams.*
- virtual void **run** ()
 

*Runs the polling thread.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeid)
 

*Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
 

*Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
 

*Is the **Transport** (p. 3819) Connected to its Broker.*
- virtual bool **isClosed** () const

*Has the **Transport** (p. 3819) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP\_UNUSED) throw ( **decaf::io::IOException** )

*reconnect to another location*

### 6.424.1 Detailed Description

Implementation of the **Transport** (p. 3819) interface that performs marshaling of commands to IO streams.

This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming commands. When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

### 6.424.2 Constructor & Destructor Documentation

#### 6.424.2.1 **activemq::transport::IOTransport::IOTransport** ( )

Default Constructor.

#### 6.424.2.2 **activemq::transport::IOTransport::IOTransport** ( const **Pointer**< **wireformat::WireFormat** > & *wireFormat* )

Create an instance of this **Transport** (p. 3819) and assign its WireFormat instance at creation time.

#### Parameters

<i>wireFormat</i>	Data encoder / decoder to use when reading and writing.
-------------------	---

#### 6.424.2.3 **virtual activemq::transport::IOTransport::~~IOTransport** ( ) [virtual]

### 6.424.3 Member Function Documentation

#### 6.424.3.1 **virtual void activemq::transport::IOTransport::close** ( ) throw ( **decaf::io::IOException** ) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

**Exceptions**

<i>IOException</i>	if errors occur.
--------------------	------------------

Implements **decaf::io::Closeable** (p. 1121).

6.424.3.2 `virtual std::string activemq::transport::IOTransport::getRemoteAddress ( ) const`  
`[inline, virtual]`

**Returns**

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3821).

6.424.3.3 `virtual TransportListener* activemq::transport::IOTransport::getTransportListener`  
`( ) const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

**Returns**

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3821).

6.424.3.4 `virtual bool activemq::transport::IOTransport::isClosed ( ) const` `[inline,`  
`virtual]`

Has the **Transport** (p. 3819) been shutdown and no longer usable.

**Returns**

true if the **Transport** (p. 3819)

Implements **activemq::transport::Transport** (p. 3821).

6.424.3.5 `virtual bool activemq::transport::IOTransport::isConnected ( ) const` `[inline,`  
`virtual]`

Is the **Transport** (p. 3819) Connected to its Broker.

**Returns**

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3821).

6.424.3.6 `virtual bool activemq::transport::IOTransport::isFaultTolerant ( ) const`  
`[inline, virtual]`

Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns

true if the **Transport** (p. 3819) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3822).

6.424.3.7 `virtual Transport* activemq::transport::IOTransport::narrow ( const std::type_info & typed )`  
`[inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.

#### Parameters

<i>typed</i>	- The <code>type_info</code> of the Object we are searching for.
--------------	--

#### Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3822).

6.424.3.8 `virtual void activemq::transport::IOTransport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
`[virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3822).

6.424.3.9 `virtual void activemq::transport::IOTransport::reconnect ( const decaf::net::URI &uri AMQCPP_UNUSED ) throw ( decaf::io::IOException ) [inline, virtual]`

reconnect to another location

#### Parameters

<i>uri</i>	
------------	--

#### Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

6.424.3.10 `virtual Pointer<Response> activemq::transport::IOTransport::request ( const Pointer< Command > &command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Returns

the response to the command sent.

#### Exceptions

<i>UnsupportedOperationException.</i>	
---------------------------------------	--

Implements **activemq::transport::Transport** (p. 3823).

6.424.3.11 `virtual Pointer<Response> activemq::transport::IOTransport::request ( const Pointer< Command > &command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	the command to be sent.
<i>timeout</i>	the time to wait for a response.

**Returns**

the response to the command sent.

**Exceptions**

<i>UnsupportedOperationException.</i>
---------------------------------------

Implements **activemq::transport::Transport** (p. 3824).

6.424.3.12 `virtual void activemq::transport::IOTransport::run ( ) [virtual]`

Runs the polling thread.

Implements **decaf::lang::Runnable** (p. 3265).

6.424.3.13 `virtual void activemq::transport::IOTransport::setInputStream ( decaf::io::DataInputStream * is ) [inline, virtual]`

Sets the input stream for in-coming commands.

**Parameters**

<i>is</i>	The input stream.
-----------	-------------------

6.424.3.14 `virtual void activemq::transport::IOTransport::setOutputStream ( decaf::io::DataOutputStream * os ) [inline, virtual]`

Sets the output stream for out-going commands.

**Parameters**

<i>os</i>	The output stream.
-----------	--------------------

6.424.3.15 `virtual void activemq::transport::IOTransport::setTransportListener ( TransportListener * listener ) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

**Parameters**

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 3824).

6.424.3.16 virtual void activemq::transport::IOTransport::setWireFormat ( const Pointer< wireformat::WireFormat > & wireFormat ) [inline, virtual]

Sets the WireFormat instance to use.

#### Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implements **activemq::transport::Transport** (p. 3824).

6.424.3.17 virtual void activemq::transport::IOTransport::start ( ) throw ( decaf::io::IOException ) [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

#### Exceptions

<i>CMSEException</i>	if an error occurs or if this transport has already been closed.
----------------------	--

Implements **activemq::transport::Transport** (p. 3825).

6.424.3.18 virtual void activemq::transport::IOTransport::stop ( ) throw ( decaf::io::IOException ) [virtual]

Stops the **Transport** (p. 3819), terminating any threads and stopping all read and write operations.

#### Exceptions

<i>IOException</i>	if an error occurs while stopping the <b>Transport</b> (p. 3819).
--------------------	---

Implements **activemq::transport::Transport** (p. 3825).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/IOTransport.h

## 6.425 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 2112) type for generic collections API calls.

```
#include <src/main/decaf/lang/Iterable.h>
```

Inheritance diagram for `decaf::lang::Iterable< E >`:

## Public Member Functions

- virtual `~Iterable()`
- virtual `decaf::util::Iterator< E > * iterator()=0`
- virtual `decaf::util::Iterator< E > * iterator() const =0`

### 6.425.1 Detailed Description

`template<typename E>class decaf::lang::Iterable< E >`

Implementing this interface allows an object to be cast to an **Iterable** (p.2112) type for generic collections API calls.

### 6.425.2 Constructor & Destructor Documentation

6.425.2.1 `template<typename E> virtual decaf::lang::Iterable< E >::~~Iterable()`  
`[inline, virtual]`

### 6.425.3 Member Function Documentation

6.425.3.1 `template<typename E> virtual decaf::util::Iterator<E>* decaf::lang::Iterable< E >::iterator()` `[pure virtual]`

## Returns

an iterator over a set of elements of type T.

Implemented in `decaf::util::concurrent::SynchronousQueue< E >` (p.3665), `decaf::util::PriorityQueue< E >` (p.2980), `decaf::util::StlList< E >` (p.3539), `decaf::util::StlSet< E >` (p.3570), `decaf::util::StlList< cms::MessageConsumer * >` (p.3539), `decaf::util::StlList< CompositeTask * >` (p.3539), `decaf::util::StlList< URI >` (p.3539), `decaf::util::StlList< Pointer< DestinationInfo > >` (p.3539), `decaf::util::StlList< PrimitiveValueNode >` (p.3539), `decaf::util::StlList< Pointer< Command > >` (p.3539), `decaf::util::StlList< Pointer< BackupTransport > >` (p.3539), `decaf::util::StlList< cms::MessageProducer * >` (p.3539), `decaf::util::StlList< cms::Destination * >` (p.3539), `decaf::util::StlList< cms::Session * >` (p.3539), `decaf::util::StlList< cms::Connection * >` (p.3539), `decaf::util::StlSet< transport::TransportListener * >` (p.3570), `decaf::util::StlSet< Pointer< Synchronization > >` (p.3570), `decaf::util::StlSet< Resource * >` (p.3570), and `decaf::util::StlSet< ActiveMQSession * >` (p.3570).

Referenced by `decaf::util::AbstractCollection< cms::Connection * >::clear()`, `decaf::util::AbstractCollection< cms::Connection * >::contains()`, `decaf::util::AbstractCollection< cms::Connection * >::copy()`, `decaf::util::AbstractCollection< cms::Connection * >::operator=()`, `decaf::util::AbstractCollection<`



cms::Connection \* >::remove(), decaf::util::AbstractSet< ActiveMQSession \* >::removeAll(), decaf::util::AbstractCollection< cms::Connection \* >::removeAll(), decaf::util::AbstractCollection< cms::Connection \* >::retainAll(), and decaf::util::AbstractCollection< cms::Connection \* >::toArray().

6.425.3.2 `template<typename E> virtual decaf::util::Iterator<E>*`  
`decaf::lang::Iterable< E >::iterator ( ) const [pure virtual]`

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3665), **decaf::util::PriorityQueue< E >** (p. 2980), **decaf::util::StlList< E >** (p. 3539), **decaf::util::StlSet< E >** (p. 3570), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3539), **decaf::util::StlList< CompositeTask \* >** (p. 3539), **decaf::util::StlList< URI >** (p. 3539), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3539), **decaf::util::StlList< PrimitiveValueNode >** (p. 3539), **decaf::util::StlList< Pointer< Command > >** (p. 3539), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3539), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3539), **decaf::util::StlList< cms::Destination \* >** (p. 3539), **decaf::util::StlList< cms::Session \* >** (p. 3539), **decaf::util::StlList< cms::Connection \* >** (p. 3539), **decaf::util::StlSet< transport::TransportListener \* >** (p. 3570), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 3570), **decaf::util::StlSet< Resource \* >** (p. 3570), and **decaf::util::StlSet< ActiveMQSession \* >** (p. 3570).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Iterable.h

## 6.426 decaf::util::Iterator< T > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

```
#include <src/main/decaf/util/Iterator.h>
```

Inheritance diagram for decaf::util::Iterator< T >:

### Public Member Functions

- virtual **~Iterator** ()
- virtual T **next** ()=0 throw ( lang::exceptions::NoSuchElementException )  
*Returns the next element in the iteration.*
- virtual bool **hasNext** () const =0  
*Returns true if the iteration has more elements.*
- virtual void **remove** ()=0 throw ( lang::exceptions::IllegalStateException, lang::exceptions::UnsupportedOperationException )  
*Removes from the underlying collection the last element returned by the iterator (optional operation).*

### 6.426.1 Detailed Description

```
template<typename T>class decaf::util::Iterator< T >
```

Defines an object that can be used to iterate over the elements of a collection.

The iterator provides a way to access and remove elements with well defined semantics.

### 6.426.2 Constructor & Destructor Documentation

```
6.426.2.1 template<typename T> virtual decaf::util::Iterator< T >::~~Iterator ( )
[inline, virtual]
```

### 6.426.3 Member Function Documentation

```
6.426.3.1 template<typename T> virtual bool decaf::util::Iterator< T >::hasNext ( ) const
[pure virtual]
```

Returns true if the iteration has more elements.

Returns false if the next call to next would result in an NoSuchElementException to be thrown.

```
6.426.3.2 template<typename T> virtual T decaf::util::Iterator< T >::next ( ) throw (
lang::exceptions::NoSuchElementException ) [pure virtual]
```

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p.2115) method returns false will return each element in the underlying collection exactly once.

#### Returns

next element in the iteration of elements

#### Exceptions

<i>NoSuchElementException</i>	- iteration has no more elements.
-------------------------------	-----------------------------------

```
6.426.3.3 template<typename T> virtual void decaf::util::Iterator< T
>::remove ( ) throw ( lang::exceptions::IllegalStateException,
lang::exceptions::UnsupportedOperationException ) [pure
virtual]
```

Removes from the underlying collection the last element returned by the iterator (optional operation).

This method can be called only once per call to next. The behavior of an iterator is

unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

### Exceptions

<i>UnsupportedOperationException</i>	- if the remove operation is not supported by this <b>Iterator</b> (p. 2114).
<i>IllegalStateException</i>	- if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

The documentation for this class was generated from the following file:

- src/main/decaf/util/Iterator.h

## 6.427 activemq::commands::JournalQueueAck Class Reference

```
#include <src/main/activemq/commands/JournalQueueAck.h>
```

Inheritance diagram for activemq::commands::JournalQueueAck:

### Public Member Functions

- **JournalQueueAck** ()
- virtual **~JournalQueueAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalQueueAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageAck** > & **getMessageAck** () const
- virtual **Pointer**< **MessageAck** > & **getMessageAck** ()
- virtual void **setMessageAck** (const **Pointer**< **MessageAck** > &messageAck)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALQUEUEACK** = 52

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageAck** > **messageAck**

### 6.427.1 Constructor & Destructor Documentation

6.427.1.1 `activemq::commands::JournalQueueAck::JournalQueueAck ( )`

6.427.1.2 `virtual activemq::commands::JournalQueueAck::~~JournalQueueAck ( )`  
[virtual]

### 6.427.2 Member Function Documentation

6.427.2.1 `virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.427.2.2 `virtual void activemq::commands::JournalQueueAck::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.427.2.3 `virtual bool activemq::commands::JournalQueueAck::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.427.2.4 `virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.427.2.5 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ( ) const` [virtual]

6.427.2.6 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ( )` [virtual]

6.427.2.7 `virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ( ) const` [virtual]

6.427.2.8 `virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ( )` [virtual]

6.427.2.9 `virtual void activemq::commands::JournalQueueAck::setDestination ( const Pointer< ActiveMQDestination > & destination )` [virtual]

6.427.2.10 `virtual void activemq::commands::JournalQueueAck::setMessageAck ( const Pointer< MessageAck > & messageAck )` [virtual]

6.427.2.11 `virtual std::string activemq::commands::JournalQueueAck::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

### 6.427.3 Field Documentation

6.427.3.1 `Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination`  
`[protected]`

6.427.3.2 `const unsigned char activemq::commands::JournalQueueAck::ID_ - JOURNALQUEUEACK = 52` `[static]`

6.427.3.3 `Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck`  
`[protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalQueueAck.h`

## 6.428 `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2119).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAck
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller`:

### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*

6.428

activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller

Class Reference

2129

- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.428.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2119).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.428.2 Constructor & Destructor Documentation

6.428.2.1 **activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.428.2.2 **virtual activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.428.3 Member Function Documentation

6.428.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.428.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.428.3.3 virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.428.3.4 virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source



6.428

**activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller**

**Class Reference**

**2131**

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.428.3.5 virtual int activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.428.3.6 virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.428.3.7 virtual void activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalQueueAckMarshaller.h**

## 6.429 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2123).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**:

#### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

6.429

**activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

**Class Reference**

**2133**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.429.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2123).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.429.2 Constructor & Destructor Documentation

6.429.2.1 **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.429.2.2 **virtual activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.429.3 Member Function Documentation

6.429.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.429.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.429.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.429.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.429

**activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

**Class Reference**

2135

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.429.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.429.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.429.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalQueueAckMarshaller.h**

## 6.430 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2127).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**:

#### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

## 6.430

**activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

### Class Reference

2137

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.430.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2127).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.430.2 Constructor & Destructor Documentation

6.430.2.1 **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.430.2.2 **virtual activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.430.3 Member Function Documentation

6.430.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.430.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.430.3.3  virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.430.3.4  virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



6.430

**activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

**Class Reference**

2139

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.430.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.430.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.430.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalQueueAckMarshaller.h**

## 6.431 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2131).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**:

#### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

## 6.431

**activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

### Class Reference

2141

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.431.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2131).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.431.2 Constructor & Destructor Documentation

6.431.2.1 **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.431.2.2 **virtual activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.431.3 Member Function Documentation

6.431.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.431.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.431.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.431.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.431

**activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

**Class Reference**

**2143**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.431.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.431.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.431.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalQueueAckMarshaller.h**

## 6.432 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2135).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAck
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**:

#### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

## 6.432

**activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

### Class Reference

2145

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.432.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2135).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.432.2 Constructor & Destructor Documentation

6.432.2.1 **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.432.2.2 **virtual activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.432.3 Member Function Documentation

6.432.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.432.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.432.3.3  virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.432.3.4  virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



6.432

**activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

**Class Reference**

2147

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.432.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.432.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.432.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalQueueAckMarshaller.h**

## 6.433 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2139).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAck
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**:

#### Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

## 6.433

**activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

### Class Reference

2149

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.433.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2139).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.433.2 Constructor & Destructor Documentation

6.433.2.1 **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::JournalQueueAckMarshaller**  
( ) [inline]

6.433.2.2 **virtual activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller**  
( ) [inline, virtual]

### 6.433.3 Member Function Documentation

6.433.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.433.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.433.3.3 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.433.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.433

**activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

**Class Reference**

2151

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.433.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.433.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.433.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalQueueAckMarshaller.h**

## 6.434 activemq::commands::JournalTopicAck Class Reference

```
#include <src/main/activemq/commands/JournalTopicAck.h>
```

Inheritance diagram for **activemq::commands::JournalTopicAck**:

#### Public Member Functions

- **JournalTopicAck** ()
- virtual **~JournalTopicAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalTopicAck** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual long long **getMessageSequenceId** () const
- virtual void **setMessageSequenceId** (long long messageSequenceId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALTOPICACK** = 50

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **MessageId** > messageId
- long long messageSequenceId
- std::string subscriptionName
- std::string clientId
- **Pointer**< **TransactionId** > transactionId

### 6.434.1 Constructor & Destructor Documentation

6.434.1.1 **activemq::commands::JournalTopicAck::JournalTopicAck** ( )

6.434.1.2 **virtual activemq::commands::JournalTopicAck::~~JournalTopicAck** ( )  
[virtual]

### 6.434.2 Member Function Documentation

6.434.2.1 `virtual JournalTopicAck* activemq::commands::JournalTopicAck::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.434.2.2 `virtual void activemq::commands::JournalTopicAck::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.434.2.3 `virtual bool activemq::commands::JournalTopicAck::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.434.2.4 `virtual const std::string& activemq::commands::JournalTopicAck::getClientId ( ) const [virtual]`

6.434.2.5 `virtual std::string& activemq::commands::JournalTopicAck::getClientId ( ) [virtual]`

6.434.2.6 `virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.



**Returns**

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

- 6.434.2.7 virtual const **Pointer**<**ActiveMQDestination**>& activemq::commands::JournalTopicAck::getDestination ( ) const [virtual]
- 6.434.2.8 virtual **Pointer**<**ActiveMQDestination**>& activemq::commands::JournalTopicAck::getDestination ( ) [virtual]
- 6.434.2.9 virtual const **Pointer**<**MessageId**>& activemq::commands::JournalTopicAck::getMessageId ( ) const [virtual]
- 6.434.2.10 virtual **Pointer**<**MessageId**>& activemq::commands::JournalTopicAck::getMessageId ( ) [virtual]
- 6.434.2.11 virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId ( ) const [virtual]
- 6.434.2.12 virtual std::string& activemq::commands::JournalTopicAck::getSubscriptionName ( ) [virtual]
- 6.434.2.13 virtual const std::string& activemq::commands::JournalTopicAck::getSubscriptionName ( ) const [virtual]
- 6.434.2.14 virtual const **Pointer**<**TransactionId**>& activemq::commands::JournalTopicAck::getTransactionId ( ) const [virtual]
- 6.434.2.15 virtual **Pointer**<**TransactionId**>& activemq::commands::JournalTopicAck::getTransactionId ( ) [virtual]
- 6.434.2.16 virtual void activemq::commands::JournalTopicAck::setClientId ( const std::string & *clientId* ) [virtual]
- 6.434.2.17 virtual void activemq::commands::JournalTopicAck::setDestination ( const **Pointer**< **ActiveMQDestination** > & *destination* ) [virtual]
- 6.434.2.18 virtual void activemq::commands::JournalTopicAck::setMessageId ( const **Pointer**< **MessageId** > & *messageId* ) [virtual]
- 6.434.2.19 virtual void activemq::commands::JournalTopicAck::setMessageSequenceId ( long long *messageSequenceId* ) [virtual]

6.434.2.20 `virtual void activemq::commands::JournalTopicAck::setSubscriptionName ( const std::string & subscriptionName ) [virtual]`

6.434.2.21 `virtual void activemq::commands::JournalTopicAck::setTransactionId ( const Pointer< TransactionId > & transactionId ) [virtual]`

6.434.2.22 `virtual std::string activemq::commands::JournalTopicAck::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

### 6.434.3 Field Documentation

6.434.3.1 `std::string activemq::commands::JournalTopicAck::clientId [protected]`

6.434.3.2 `Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination [protected]`

6.434.3.3 `const unsigned char activemq::commands::JournalTopicAck::ID_- JOURNALTOPICACK = 50 [static]`

6.434.3.4 `Pointer<MessageId> activemq::commands::JournalTopicAck::messageId [protected]`

6.434.3.5 `long long activemq::commands::JournalTopicAck::messageSequenceId [protected]`

6.434.3.6 `std::string activemq::commands::JournalTopicAck::subscriptionName [protected]`

6.434.3.7 `Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTopicAck.h`

## 6.435 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2148).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.435.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2148).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.435.2 Constructor & Destructor Documentation

6.435.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.435.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

### 6.435.3 Member Function Documentation

6.435.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.435.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.435.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.435.3.4 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseUnmarshal  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataInputStream** \* *dataIn* ) throw ( **decaf::io::IOException** )  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.435.3.5 virtual int activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal1  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.435.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.435.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTopicAckMarshaller.h**

## 6.436 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2152).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.436.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2152).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.436.2 Constructor & Destructor Documentation

6.436.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.436.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

## 6.436.3 Member Function Documentation

6.436.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.436.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.436.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.436.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.436.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.436.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.436.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTopicAckMarshaller.h**

## 6.437 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2156).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.437.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2156).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.437.2 Constructor & Destructor Documentation

6.437.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.437.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

## 6.437.3 Member Function Documentation

6.437.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.437.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.437.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.437.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.437.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.437.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.437.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTopicAckMarshaller.h**

## 6.438 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2160).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.438.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2160).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.438.2 Constructor & Destructor Documentation

6.438.2.1 `activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.438.2.2 `virtual activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

## 6.438.3 Member Function Documentation

6.438.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.438.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.438.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.438.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.438.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.438.3.6 virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.438.3.7 virtual void activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalTopicAckMarshaller.h**

## 6.439 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2164).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.439.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2164).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.439.2 Constructor & Destructor Documentation

6.439.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.439.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

## 6.439.3 Member Function Documentation

6.439.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.439.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.439.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.439.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.439.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```

6.439.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```

6.439.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTopicAckMarshaller.h**

## 6.440 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2168).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller:

### Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.440.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2168).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.440.2 Constructor & Destructor Documentation

6.440.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::JournalTopicAckMarshaller ( ) [inline]`

6.440.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller ( ) [inline, virtual]`

## 6.440.3 Member Function Documentation

6.440.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.440.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.440.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.440.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.440.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.440.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.440.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h`

## 6.441 activemq::commands::JournalTrace Class Reference

```
#include <src/main/activemq/commands/JournalTrace.h>
```

Inheritance diagram for activemq::commands::JournalTrace:

## Public Member Functions

- **JournalTrace** ()
- virtual **~JournalTrace** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalTrace \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &message)

## Static Public Attributes

- static const unsigned char **ID\_JOURNALTRACE** = 53

## Protected Attributes

- std::string **message**

## 6.441.1 Constructor & Destructor Documentation

6.441.1.1 **activemq::commands::JournalTrace::JournalTrace** ( )

6.441.1.2 **virtual activemq::commands::JournalTrace::~~JournalTrace** ( ) [virtual]

## 6.441.2 Member Function Documentation

6.441.2.1 `virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure ( )`  
`const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.441.2.2 `virtual void activemq::commands::JournalTrace::copyDataStructure ( const DataStructure * src )` `[virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.441.2.3 `virtual bool activemq::commands::JournalTrace::equals ( const DataStructure * value ) const` `[virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.441.2.4 `virtual unsigned char activemq::commands::JournalTrace::getDataStructureType ( )`  
`const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

## 6.442 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller

### Class Reference 2183

---

- 6.441.2.5 `virtual std::string& activemq::commands::JournalTrace::getMessage ( )`  
[virtual]
- 6.441.2.6 `virtual const std::string& activemq::commands::JournalTrace::getMessage ( ) const`  
[virtual]
- 6.441.2.7 `virtual void activemq::commands::JournalTrace::setMessage ( const std::string & message )` [virtual]
- 6.441.2.8 `virtual std::string activemq::commands::JournalTrace::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 796).

#### 6.441.3 Field Documentation

- 6.441.3.1 `const unsigned char activemq::commands::JournalTrace::ID_ - JOURNALTRACE = 53` [static]
- 6.441.3.2 `std::string activemq::commands::JournalTrace::message`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTrace.h`

## 6.442 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller

### Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2174).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller`:

#### Public Member Functions

- **JournalTraceMarshaller** ()

- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.442.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2174).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.442.2 Constructor & Destructor Documentation

6.442.2.1 **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::JournalTraceMarshaller** ( ) [inline]

6.442.2.2 **virtual activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::~~JournalTraceMarshaller** ( ) [inline, virtual]

### 6.442.3 Member Function Documentation

6.442.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.442.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::getDataStructureType  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.442.3.3 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseMarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \*  
 dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.442.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseUnmarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,  
 decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.442.3.5  virtual int activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.442.3.6  virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).



## 6.443 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller

### Class Reference 2187

6.442.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTraceMarshaller.h**

## 6.443 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller

### Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2178).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller:

#### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.443.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2178).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.443.2 Constructor & Destructor Documentation

6.443.2.1 **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::JournalTraceMarshaller**  
( ) [inline]

6.443.2.2 **virtual activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::~~JournalTraceMarshaller**  
( ) [inline, virtual]

### 6.443.3 Member Function Documentation

6.443.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.443.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.443.3.3 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.443.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.443.3.5  virtual int activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.443.3.6  virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

## 6.444 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller

### Class Reference 2191

6.443.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTraceMarshaller.h**

## 6.444 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller

### Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2182).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller:

#### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

#### 6.444.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2182).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

#### 6.444.2 Constructor & Destructor Documentation

6.444.2.1 **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::JournalTraceMarshaller**  
( ) [inline]

6.444.2.2 **virtual activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::~~JournalTraceMarshaller**  
( ) [inline, virtual]

#### 6.444.3 Member Function Documentation

6.444.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.444.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.444.3.3 virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.444.3.4 virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.444.3.5  virtual int activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.444.3.6  virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).



## 6.445 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller

### Class Reference 2195

6.444.3.7 virtual void activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalTraceMarshaller.h**

## 6.445 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller

### Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2186).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller:

#### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.445.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2186).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.445.2 Constructor & Destructor Documentation

6.445.2.1 **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::JournalTraceMarshaller**  
( ) [inline]

6.445.2.2 **virtual activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::~~JournalTraceMarshaller**  
( ) [inline, virtual]

### 6.445.3 Member Function Documentation

6.445.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.445.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.445.3.3 virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.445.3.4 virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.445.3.5  virtual int activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.445.3.6  virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.445.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTraceMarshaller.h**

## 6.446 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2190).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller:

#### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.446.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2190).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.446.2 Constructor & Destructor Documentation

6.446.2.1 **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::JournalTraceMarshaller**  
( ) [inline]

6.446.2.2 **virtual activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::~~JournalTraceMarshaller**  
( ) [inline, virtual]

### 6.446.3 Member Function Documentation

6.446.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.446.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::getDataStructureType  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.446.3.3 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseMarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \*  
 dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.446.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseUnmarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,  
 decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.446.3.5  virtual int activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.446.3.6  virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).



## 6.447 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller

### Class Reference 2203

6.446.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTraceMarshaller.h**

## 6.447 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller

### Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2194).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller:

#### Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.447.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2194).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.447.2 Constructor & Destructor Documentation

6.447.2.1 **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::JournalTraceMarshaller**  
( ) [inline]

6.447.2.2 **virtual activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::~~JournalTraceMarshaller**  
( ) [inline, virtual]

### 6.447.3 Member Function Documentation

6.447.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.447.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.447.3.3 virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.447.3.4 virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.447.3.5  virtual int activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.447.3.6  virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.447.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTraceMarshaller.h**

## 6.448 activemq::commands::JournalTransaction Class Reference

```
#include <src/main/activemq/commands/JournalTransaction.h>
```

Inheritance diagram for activemq::commands::JournalTransaction:

#### Public Member Functions

- **JournalTransaction** ()
- virtual **~JournalTransaction** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **JournalTransaction** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*

- virtual bool **equals** (const **DataSet** \*value) const  
*Compares the **DataSet** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **getWasPrepared** () const
- virtual void **setWasPrepared** (bool wasPrepared)

### Static Public Attributes

- static const unsigned char **ID\_JOURNALTRANSACTION** = 54

### Protected Attributes

- **Pointer**< **TransactionId** > transactionId
- unsigned char type
- bool wasPrepared

## 6.448.1 Constructor & Destructor Documentation

6.448.1.1 `activemq::commands::JournalTransaction::JournalTransaction ( )`

6.448.1.2 `virtual activemq::commands::JournalTransaction::~~JournalTransaction ( )`  
[virtual]

## 6.448.2 Member Function Documentation

6.448.2.1 `virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataSet ( )`  
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataSet` (p. 1628).

6.448.2.2 `virtual void activemq::commands::JournalTransaction::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.448.2.3 `virtual bool activemq::commands::JournalTransaction::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.448.2.4 `virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.448.2.5 `virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId ( ) const [virtual]`

6.448.2.6 `virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId ( ) [virtual]`

6.448.2.7 `virtual unsigned char activemq::commands::JournalTransaction::getType ( ) const [virtual]`

- 6.448.2.8 `virtual bool activemq::commands::JournalTransaction::getWasPrepared ( ) const`  
[virtual]
- 6.448.2.9 `virtual void activemq::commands::JournalTransaction::setTransactionId ( const`  
`Pointer< TransactionId > & transactionId )` [virtual]
- 6.448.2.10 `virtual void activemq::commands::JournalTransaction::setType ( unsigned char type`  
`)` [virtual]
- 6.448.2.11 `virtual void activemq::commands::JournalTransaction::setWasPrepared ( bool`  
`wasPrepared )` [virtual]
- 6.448.2.12 `virtual std::string activemq::commands::JournalTransaction::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

### 6.448.3 Field Documentation

- 6.448.3.1 `const unsigned char activemq::commands::JournalTransaction::ID_`  
`JOURNALTRANSACTION = 54` [static]
- 6.448.3.2 `Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId`  
[protected]
- 6.448.3.3 `unsigned char activemq::commands::JournalTransaction::type`  
[protected]
- 6.448.3.4 `bool activemq::commands::JournalTransaction::wasPrepared`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTransaction.h`

## 6.449 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2201).



6.449

activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller

Class Reference

2211

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarsha
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller:

## Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.449.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2201).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.449.2 Constructor & Destructor Documentation

6.449.2.1 **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.449.2.2 `virtual activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::~~JournalTransactionMarshaller ( ) [inline, virtual]`

### 6.449.3 Member Function Documentation

6.449.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.449.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.449.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.449

**activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller**

**Class Reference**

**2213**

```
6.449.3.4 virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.449.3.5 virtual int activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.449.3.6 virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.449.3.7 virtual void activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**JournalTransactionMarshaller.h**

## 6.450 **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2205).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**:

6.450

activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller

Class Reference

2215

## Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.450.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2205).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.450.2 Constructor & Destructor Documentation

6.450.2.1 **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.450.2.2 **virtual activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.450.3 Member Function Documentation

6.450.3.1 **virtual** **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::createObject**  
**( ) const** **[virtual]**

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.450.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::getDataStructure**  
**( ) const** **[virtual]**

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.450.3.3 **virtual void** **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException )** **[virtual]**

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.450

**activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

#### Class Reference

2217

```
6.450.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseUnmarshal  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )  
    [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.450.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal1  
    ( OpenWireFormat * wireFormat, commands::DataStructure *  
      dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
    [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.450.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal2  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw  
    ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.450.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTransactionMarshaller.h**

## 6.451 **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2209).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**:



6.451

activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller

Class Reference

2219

## Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.451.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2209).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.451.2 Constructor & Destructor Documentation

6.451.2.1 **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.451.2.2 **virtual activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.451.3 Member Function Documentation

6.451.3.1 **virtual** **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::createObject**  
**( ) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.451.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::getDataStructure**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.451.3.3 **virtual void** **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException )** [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.451

**activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

**Class Reference**

**2221**

```
6.451.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.451.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.451.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.451.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTransactionMarshaller.h**

## 6.452 **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2213).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**:

6.452

activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller

Class Reference

2223

## Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.452.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2213).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.452.2 Constructor & Destructor Documentation

6.452.2.1 **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.452.2.2 **virtual activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.452.3 Member Function Documentation

6.452.3.1 **virtual** **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::createObject**  
**( ) const** **[virtual]**

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.452.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::getDataStructure**  
**( ) const** **[virtual]**

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.452.3.3 **virtual void** **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException )** **[virtual]**

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.452

**activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

## Class Reference

2225

```
6.452.3.4 virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseUnmarshal  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )  
    [virtual]
```

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.452.3.5 virtual int activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal1  
    ( OpenWireFormat * wireFormat, commands::DataStructure *  
      dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
    [virtual]
```

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.452.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal2  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw  
    ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.452.3.7 virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTransactionMarshaller.h**

## 6.453 **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2217).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**:



6.453

activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller

Class Reference

2227

## Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.453.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2217).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.453.2 Constructor & Destructor Documentation

6.453.2.1 **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.453.2.2 **virtual activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.453.3 Member Function Documentation

6.453.3.1 **virtual** **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::createObject**  
**( ) const** *[virtual]*

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.453.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::getDataStructure**  
**( ) const** *[virtual]*

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.453.3.3 **virtual void** **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException )** *[virtual]*

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.453

**activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

**Class Reference**

**2229**

```
6.453.3.4 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseUnmarshal  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )  
    [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.453.3.5 virtual int activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal1  
    ( OpenWireFormat * wireFormat, commands::DataStructure *  
      dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )  
    [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.453.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal2  
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,  
      decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw  
    ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.453.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTransactionMarshaller.h**

## 6.454 **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2221).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTransact
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**:

6.454

activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller

Class Reference

2231

## Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.454.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2221).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.454.2 Constructor & Destructor Documentation

6.454.2.1 **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::JournalTransactionMarshaller**  
( ) [inline]

6.454.2.2 **virtual activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::~~JournalTransactionMarshaller**  
( ) [inline, virtual]

### 6.454.3 Member Function Documentation

6.454.3.1 **virtual** **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::createObject**  
**( ) const** **[virtual]**

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.454.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::getDataStructure**  
**( ) const** **[virtual]**

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.454.3.3 **virtual void** **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseMarshal**  
**( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException )** **[virtual]**

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.454

**activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

**Class Reference**

**2233**

```
6.454.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.454.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.454.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.454.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTransactionMarshaller.h**

## 6.455 activemq::commands::KeepAliveInfo Class Reference

```
#include <src/main/activemq/commands/KeepAliveInfo.h>
```

Inheritance diagram for **activemq::commands::KeepAliveInfo**:



## Public Member Functions

- **KeepAliveInfo** ()
- virtual **~KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **KeepAliveInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_KEEPAALIVEINFO** = 10

## 6.455.1 Constructor & Destructor Documentation

6.455.1.1 **activemq::commands::KeepAliveInfo::KeepAliveInfo** ( )

6.455.1.2 **virtual activemq::commands::KeepAliveInfo::~~KeepAliveInfo** ( ) [virtual]

## 6.455.2 Member Function Documentation

6.455.2.1 **virtual KeepAliveInfo\*** **activemq::commands::KeepAliveInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.455.2.2 `virtual void activemq::commands::KeepAliveInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.455.2.3 `virtual bool activemq::commands::KeepAliveInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.455.2.4 `virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.455.2.5 `virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isKeepAliveInfo()** (p. 2227) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 727).

## 6.456 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller

### Class Reference 2237

---

6.455.2.6 `virtual std::string activemq::commands::KeepAliveInfo::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.455.2.7 `virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit`  
`( activemq::state::CommandVisitor * visitor ) throw (`  
`exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.455.3 Field Documentation

6.455.3.1 `const unsigned char activemq::commands::KeepAliveInfo::ID_`  
`KEEPALIVEINFO = 10 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/KeepAliveInfo.h`

## 6.456 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2228).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller**:

## Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.456.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2228).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.456.2 Constructor & Destructor Documentation

6.456.2.1 **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller** ( ) [inline]

6.456.2.2 **virtual activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller** ( ) [inline, virtual]

### 6.456.3 Member Function Documentation

6.456.3.1 virtual **commands::DataStructure\*** **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::createObject ( )**  
const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.456.3.2 virtual unsigned char **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::getDataStructureType ( )** const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.456.3.3 virtual void **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut )** throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

```
6.456.3.4 virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.456.3.5 virtual int activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

6.456.3.6 virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
 ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
 (p. 762).

6.456.3.7 virtual void activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
 (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h

## 6.457 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2233).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMa
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.457.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2233).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.457.2 Constructor & Destructor Documentation

6.457.2.1 `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]`

6.457.2.2 `virtual activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( ) [inline, virtual]`

## 6.457.3 Member Function Documentation

6.457.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.457.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.457.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.457.3.4  virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.457.3.5  virtual int activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.457 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller

---

Class Reference2245

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

6.457.3.6 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal2 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

6.457.3.7 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h

## 6.458 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2237).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMa
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.458.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2237).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.458.2 Constructor & Destructor Documentation

6.458.2.1 `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]`

6.458.2.2 `virtual activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( ) [inline, virtual]`

## 6.458.3 Member Function Documentation

6.458.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.458.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.458.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.458.3.4  virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.458.3.5  virtual int activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.458 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller

### Class Reference 2249

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.458.3.6 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.458.3.7 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

## 6.459 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2241).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMa
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.459.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2241).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.459.2 Constructor & Destructor Documentation

6.459.2.1 `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]`

6.459.2.2 `virtual activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( ) [inline, virtual]`

## 6.459.3 Member Function Documentation

6.459.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.459.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.459.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.459.3.4  virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.459.3.5  virtual int activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.459 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller

### Class Reference 2253

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.459.3.6 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.459.3.7 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h

## 6.460 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2245).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMa
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.460.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2245).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.460.2 Constructor & Destructor Documentation

6.460.2.1 `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]`

6.460.2.2 `virtual activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( ) [inline, virtual]`

## 6.460.3 Member Function Documentation

6.460.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.460.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.460.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.460.3.4  virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.460.3.5  virtual int activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.460 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller

### Class Reference 2257

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.460.3.6 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.460.3.7 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h

## 6.461 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2249).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMa
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller:

### Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.461.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2249).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.461.2 Constructor & Destructor Documentation

6.461.2.1 `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::KeepAliveInfoMarshaller ( ) [inline]`

6.461.2.2 `virtual activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::~~KeepAliveInfoMarshaller ( ) [inline, virtual]`

## 6.461.3 Member Function Documentation

6.461.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.461.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.461.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.461.3.4  virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.461.3.5  virtual int activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.461 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller

### Class Reference 2261

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.461.3.6  virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.461.3.7  virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h

## 6.462 decaf::security::Key Class Reference

The **Key** (p. 2253) interface is the top-level interface for all keys.

```
#include <src/main/decaf/security/Key.h>
```

Inheritance diagram for decaf::security::Key:

### Public Member Functions

- virtual **~Key** ()
- virtual std::string **getAlgorithm** () const =0  
*Returns the standard algorithm name for this key.*
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0  
*Provides the key in its primary encoding format, or nothing if this key does not support encoding.*
- virtual std::string **getFormat** () const =0  
*Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.*

### 6.462.1 Detailed Description

The **Key** (p. 2253) interface is the top-level interface for all keys.

It defines the functionality shared by all key objects. All keys have three characteristics:

An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the getAlgorithm method.

An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 SubjectPublicKeyInfo or PKCS#8), and is returned using the getEncoded method. Note: The syntax of the ASN.1 type SubjectPublicKeyInfo is defined as follows:

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm AlgorithmIdentifier,
    subjectPublicKey BIT STRING }

AlgorithmIdentifier ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER,
    parameters ANY DEFINED BY algorithm OPTIONAL }
```

For more information, see RFC 2459: Internet X.509 Public **Key** (p. 2253) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the `getFormat` method.

## 6.462.2 Constructor & Destructor Documentation

6.462.2.1 `virtual decaf::security::Key::~Key ( ) [inline, virtual]`

## 6.462.3 Member Function Documentation

6.462.3.1 `virtual std::string decaf::security::Key::getAlgorithm ( ) const [pure virtual]`

Returns the standard algorithm name for this key.

For example, "DSA" would indicate that this key is a DSA key.

### Returns

the name of the algorithm associated with this key.

6.462.3.2 `virtual void decaf::security::Key::getEncoded ( std::vector< unsigned char > & output ) const [pure virtual]`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

### Parameters

<i>output</i>	Receives the encoded key, or nothing if the key does not support encoding.
---------------	--

6.462.3.3 `virtual std::string decaf::security::Key::getFormat ( ) const [pure virtual]`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

The primary encoding format is named in terms of the appropriate ASN.1 data format, if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is `SubjectPublicKeyInfo`, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is `PrivateKeyInfo`, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

**Returns**

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Key.h**

**6.463 decaf::security::KeyException Class Reference**

```
#include <src/main/decaf/security/KeyException.h>
```

Inheritance diagram for decaf::security::KeyException:

**Public Member Functions**

- **KeyException** () throw ()  
*Default Constructor.*
- **KeyException** (const decaf::lang::Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **KeyException** (const **KeyException** &ex) throw ()  
*Copy Constructor.*
- **KeyException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **KeyException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **KeyException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **KeyException** \* **clone** () const  
*Clones this exception.*
- virtual ~**KeyException** () throw ()

**6.463.1 Constructor & Destructor Documentation****6.463.1.1 decaf::security::KeyException::KeyException ( ) throw () [inline]**

Default Constructor.

6.463.1.2 `decaf::security::KeyException::KeyException ( const decaf::lang::Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.463.1.3 `decaf::security::KeyException::KeyException ( const KeyException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.463.1.4 `decaf::security::KeyException::KeyException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.463.1.5 `decaf::security::KeyException::KeyException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.463.1.6 `decaf::security::KeyException::KeyException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
...	list of primitives that are formatted into the message

6.463.1.7 `virtual decaf::security::KeyException::~KeyException ( ) throw ()` `[inline, virtual]`

### 6.463.2 Member Function Documentation

6.463.2.1 `virtual KeyException* decaf::security::KeyException::clone ( ) const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1936).

Reimplemented in `decaf::security::InvalidKeyException` (p. 2096), and `decaf::security::KeyManagementException` (p. 2260).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyException.h`

## 6.464 decaf::security::KeyManagementException Class Reference

```
#include <src/main/decaf/security/KeyManagementException.h>
```

Inheritance diagram for `decaf::security::KeyManagementException`:



## Public Member Functions

- **KeyManagementException** () throw ()  
*Default Constructor.*
- **KeyManagementException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **KeyManagementException** (const **KeyManagementException** &ex) throw ()  
*Copy Constructor.*
- **KeyManagementException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **KeyManagementException** (const std::exception \*cause) throw ()  
*Constructor.*
- **KeyManagementException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **KeyManagementException** \* **clone** () const  
*Clones this exception.*
- virtual ~**KeyManagementException** () throw ()

## 6.464.1 Constructor & Destructor Documentation

6.464.1.1 decaf::security::KeyManagementException::KeyManagementException ( ) throw ()  
[inline]

Default Constructor.

6.464.1.2 decaf::security::KeyManagementException::KeyManagementException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.464.1.3 decaf::security::KeyManagementException::KeyManagementException ( const KeyManagementException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.464.1.4 `decaf::security::KeyManagementException::KeyManagementException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.464.1.5 `decaf::security::KeyManagementException::KeyManagementException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.464.1.6 `decaf::security::KeyManagementException::KeyManagementException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.464.1.7 `virtual decaf::security::KeyManagementException::~~KeyManagementException ( ) throw () [inline, virtual]`

## 6.464.2 Member Function Documentation

```
6.464.2.1 virtual KeyManagementException* de-
caf::security::KeyManagementException::clone ( ) const
[inline, virtual]
```

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 2257).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**KeyManagementException.h**

## 6.465 activemq::commands::LastPartialCommand Class Reference

```
#include <src/main/activemq/commands/LastPartialCommand.h>
```

Inheritance diagram for activemq::commands::LastPartialCommand:

### Public Member Functions

- **LastPartialCommand** ()
- virtual **~LastPartialCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **LastPartialCommand \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*

## Static Public Attributes

- static const unsigned char **ID\_LASTPARTIALCOMMAND** = 61

## 6.465.1 Constructor & Destructor Documentation

6.465.1.1 `activemq::commands::LastPartialCommand::LastPartialCommand ( )`

6.465.1.2 `virtual activemq::commands::LastPartialCommand::~~LastPartialCommand ( )`  
[virtual]

## 6.465.2 Member Function Documentation

6.465.2.1 `virtual LastPartialCommand* activemq::commands::LastPartialCommand::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::PartialCommand` (p. 2868).

6.465.2.2 `virtual void activemq::commands::LastPartialCommand::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from `activemq::commands::PartialCommand` (p. 2868).

6.465.2.3 `virtual bool activemq::commands::LastPartialCommand::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

6.466

**activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**

**Class Reference**

2271

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::PartialCommand** (p. 2868).

6.465.2.4 `virtual unsigned char activemq::commands::LastPartialCommand::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::PartialCommand** (p. 2869).

6.465.2.5 `virtual std::string activemq::commands::LastPartialCommand::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 2869).

### 6.465.3 Field Documentation

6.465.3.1 `const unsigned char activemq::commands::LastPartialCommand::ID_ - LASTPARTIALCOMMAND = 61 [static]`

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**LastPartialCommand.h**

## 6.466 **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2262).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**:

## Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.466.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2262).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.466.2 Constructor & Destructor Documentation

- 6.466.2.1 **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::LastPartialCommandMarshaller** ( ) [*inline*]
- 6.466.2.2 **virtual activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller** ( ) [*inline, virtual*]

### 6.466.3 Member Function Documentation

6.466

**activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**

**Class Reference**

2273

```
6.466.3.1 virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::createObject
( ) const [virtual]
```

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2871).

```
6.466.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2871).

```
6.466.3.3 virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2872).

```

6.466.3.4  virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2872).

```

6.466.3.5  virtual int activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
    [virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2873).



6.466

**activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**

**Class Reference**

**2275**

```
6.466.3.6 virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2873).

```
6.466.3.7 virtual void activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller** (p. 2874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**LastPartialCommandMarshaller.h**

## 6.467 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2267).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LastPartialComm
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.467.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2267).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.467

activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller

Class Reference

2277

## 6.467.2 Constructor & Destructor Documentation

6.467.2.1 `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::LastPartialCommandMarshaller ( ) [inline]`

6.467.2.2 `virtual activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller ( ) [inline, virtual]`

## 6.467.3 Member Function Documentation

6.467.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2884).

6.467.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2884).

6.467.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2885).

```
6.467.3.4  virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2885).

```
6.467.3.5  virtual int activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.467

**activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

**Class Reference**

2279

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2886).

6.467.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2886).

6.467.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2887).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**LastPartialCommandMarshaller.h**

## 6.468 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2271).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LastPartialComm
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.468.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2271).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.468

activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller

Class Reference

2281

## 6.468.2 Constructor & Destructor Documentation

6.468.2.1 `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::LastPartialCommandMarshaller ( ) [inline]`

6.468.2.2 `virtual activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller ( ) [inline, virtual]`

## 6.468.3 Member Function Documentation

6.468.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2875).

6.468.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2876).

6.468.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2876).

```
6.468.3.4  virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2876).

```
6.468.3.5  virtual int activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



6.468

**activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

**Class Reference**

2283

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2877).

6.468.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2877).

6.468.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2878).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LastPartialCommandMarshaller.h**

## 6.469 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2275).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LastPartialComm
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.469.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2275).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.469

activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller

Class Reference

2285

## 6.469.2 Constructor & Destructor Documentation

6.469.2.1 `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::LastPartialCommandMarshaller ( ) [inline]`

6.469.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller ( ) [inline, virtual]`

## 6.469.3 Member Function Documentation

6.469.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2880).

6.469.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2880).

6.469.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2880).

```
6.469.3.4  virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2881).

```
6.469.3.5  virtual int activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.469

**activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**

**Class Reference**

**2287**

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2881).

6.469.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2882).

6.469.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2882).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h`

## 6.470 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2279).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LastPartialComm
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.470.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2279).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.470

activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller

Class Reference

2289

## 6.470.2 Constructor & Destructor Documentation

6.470.2.1 `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::LastPartialCommandMarshaller ( ) [inline]`

6.470.2.2 `virtual activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller ( ) [inline, virtual]`

## 6.470.3 Member Function Documentation

6.470.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2888).

6.470.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2889).

6.470.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2889).

```
6.470.3.4  virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2889).

```
6.470.3.5  virtual int activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



6.470

**activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller**

**Class Reference**

2291

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2890).

6.470.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2890).

6.470.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2891).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LastPartialCommandMarshaller.h**

## 6.471 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2283).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LastPartialComm
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller:

### Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.471.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2283).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.471

**activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

**Class Reference**

**2293**

## 6.471.2 Constructor & Destructor Documentation

6.471.2.1 **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::LastPartialCommandMarshaller**  
( ) [inline]

6.471.2.2 **virtual activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller**  
( ) [inline, virtual]

## 6.471.3 Member Function Documentation

6.471.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2893).

6.471.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2893).

6.471.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2893).

```
6.471.3.4  virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2894).

```
6.471.3.5  virtual int activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.471

**activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

**Class Reference**

2295

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2894).

6.471.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2895).

6.471.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2895).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h`

## 6.472 decaf::util::comparators::Less< E > Class Template Reference

Simple **Less** (p. 2287) **Comparator** (p. 1189) that compares to elements to determine if the first is less than the second.

```
#include <src/main/decaf/util/comparators/Less.h>
```

Inheritance diagram for decaf::util::comparators::Less< E >:

### Public Member Functions

- **Less** ()
- virtual **~Less** ()
- virtual bool **operator()** (const E &left, const E &right) const  
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1189) to be passed to an STL **Map** (p. 2419) for use as the sorting criteria.*
- virtual int **compare** (const E &o1, const E &o2) const  
*Compares its two arguments for order.*

### 6.472.1 Detailed Description

```
template<typename E>class decaf::util::comparators::Less< E >
```

Simple **Less** (p. 2287) **Comparator** (p. 1189) that compares to elements to determine if the first is less than the second.

This can be used in **Collection** (p. 1155) classes to sort elements according to their natural ordering. By design the Comparator's compare function return more information about comparison than the STL binary function's boolean compare operator. In this case the compare method will return

**Since**

1.0

### 6.472.2 Constructor & Destructor Documentation

6.472.2.1 `template<typename E > decaf::util::comparators::Less< E >::Less ( )`  
`[inline]`

6.472.2.2 `template<typename E > virtual decaf::util::comparators::Less< E >::~~Less ( )`  
`[inline, virtual]`

### 6.472.3 Member Function Documentation

6.472.3.1 `template<typename E> virtual int decaf::util::comparators::Less< E  
>::compare ( const E & o1, const E & o2 ) const [inline, virtual]`

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn( compare(x, y) ) == -sgn(compare(y, x) )` for all `x` and `y`. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all `z`.

It is generally the case, but not strictly required that `(compare(x, y)==0) == ( x == y )`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

#### Parameters

<i>o1</i>	- the first object to be compared
<i>o2</i>	- the second object to be compared

#### Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implements **decaf::util::Comparator< E >** (p. 1190).

6.472.3.2 `template<typename E> virtual bool decaf::util::comparators::Less< E  
>::operator() ( const E & left, const E & right ) const [inline, virtual]`

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1189) to be passed to an STL **Map** (p. 2419) for use as the sorting criteria.

#### Parameters

<i>left</i>	- the Left hand side operand.
<i>right</i>	- the Right hand side operand.

#### Returns

true if the value of left is less than the value of right.

Implements **decaf::util::Comparator< E >** (p. 1191).

The documentation for this class was generated from the following file:

- src/main/decaf/util/comparators/**Less.h**

### 6.473 **std::less< decaf::lang::ArrayPointer< T > > Struct Template Reference**

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

```
#include <src/main/decaf/lang/ArrayPointer.h>
```

Inheritance diagram for std::less< decaf::lang::ArrayPointer< T > >:

#### Public Member Functions

- bool **operator()** (const **decaf::lang::ArrayPointer< T >** &left, const **decaf::lang::ArrayPointer< T >** &right) const

#### 6.473.1 Detailed Description

```
template<typename T>struct std::less< decaf::lang::ArrayPointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

#### 6.473.2 Member Function Documentation

6.473.2.1 **template<typename T> bool std::less< decaf::lang::ArrayPointer< T > >::operator()** ( const **decaf::lang::ArrayPointer< T >** & *left*, const **decaf::lang::ArrayPointer< T >** & *right* ) const [inline]

References **decaf::lang::ArrayPointer< T, REFCOUNTER >::get()**.

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**ArrayPointer.h**

### 6.474 **std::less< decaf::lang::Pointer< T > > Struct Template Reference**

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.



```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for `std::less< decaf::lang::Pointer< T > >`:

### Public Member Functions

- `bool operator()` (const `decaf::lang::Pointer< T >` &left, const `decaf::lang::Pointer< T >` &right) const

#### 6.474.1 Detailed Description

```
template<typename T>struct std::less< decaf::lang::Pointer< T > >
```

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

#### 6.474.2 Member Function Documentation

6.474.2.1 `template<typename T> bool std::less< decaf::lang::Pointer< T > >::operator() ( const decaf::lang::Pointer< T > & left, const decaf::lang::Pointer< T > & right ) const` [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

## 6.475 decaf::util::logging::Level Class Reference

The **Level** (p. 2290) class defines a set of standard logging levels that can be used to control logging output.

```
#include <src/main/decaf/util/logging/Level.h>
```

Inheritance diagram for `decaf::util::logging::Level`:

### Public Member Functions

- virtual `~Level ()`
- `int intValue ()` const
- `std::string getName ()` const

- `std::string toString ()` const
- virtual int **compareTo** (const **Level** &value) const
- virtual bool **equals** (const **Level** &value) const
- virtual bool **operator==** (const **Level** &value) const
- virtual bool **operator<** (const **Level** &value) const

### Static Public Member Functions

- static **Level parse** (const std::string &name) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parse a level name string into a **Level** (p. 2290).*

### Static Public Attributes

- static const **Level INHERIT**  
*NULL is a special level that indicates that the **Logger** (p. 2345) should get its **Level** (p. 2290) from its parent **Logger** (p. 2345), the value is initialized as zero.*
- static const **Level OFF**  
*OFF is a special level that can be used to turn off logging.*
- static const **Level SEVERE**  
*SEVERE is a message level indicating a serious failure.*
- static const **Level WARNING**  
*WARNING is a message level indicating a potential problem.*
- static const **Level INFO**  
*INFO is a message level for informational messages.*
- static const **Level DEBUG**  
*DEBUG is a level for more verbose informative messages.*
- static const **Level CONFIG**  
*CONFIG is a message level for static configuration messages.*
- static const **Level FINE**  
*FINE is a message level providing tracing information.*
- static const **Level FINER**  
*FINER indicates a fairly detailed tracing message.*
- static const **Level FINEST**  
*FINEST indicates a highly detailed tracing message.*
- static const **Level ALL**  
*ALL indicates that all messages should be logged.*

### Protected Member Functions

- **Level** (const std::string &name, int value)  
*Create a named **Level** (p. 2290) with a given integer value.*

### 6.475.1 Detailed Description

The **Level** (p. 2290) class defines a set of standard logging levels that can be used to control logging output.

The logging **Level** (p. 2290) objects are ordered and are specified by ordered integers. Enabling logging at a given level also enables logging at all higher levels.

Clients should normally use the predefined **Level** (p. 2290) constants such as **Level.SEVERE** (p. 2295).

The levels in descending order are:

\* SEVERE (highest value) \* WARNING \* INFO \* DEBUG \* CONFIG \* FINE \* FINER  
\* FINEST (lowest value)

In addition there is a level OFF that can be used to turn off logging, and a level ALL that can be used to enable logging of all messages.

It is possible for third parties to define additional logging levels by subclassing **Level** (p. 2290). In such cases subclasses should take care to chose unique integer level values.

#### Since

1.0

### 6.475.2 Constructor & Destructor Documentation

6.475.2.1 `decaf::util::logging::Level::Level ( const std::string & name, int value )`  
[protected]

Create a named **Level** (p. 2290) with a given integer value.

#### Parameters

<i>name</i>	Name of the level, e.g. SEVERE
<i>value</i>	Unique integer value of this level, e.g. 100

6.475.2.2 `virtual decaf::util::logging::Level::~~Level ( )` [virtual]

### 6.475.3 Member Function Documentation

6.475.3.1 `virtual int decaf::util::logging::Level::compareTo ( const Level & value ) const`  
[virtual]

6.475.3.2 `virtual bool decaf::util::logging::Level::equals ( const Level & value ) const`  
[virtual]

6.475.3.3 `std::string decaf::util::logging::Level::getName ( ) const [inline]`

#### Returns

the name of this **Level** (p. 2290) instance.

6.475.3.4 `int decaf::util::logging::Level::intValue ( ) const [inline]`

#### Returns

the integer value of this level instance.

6.475.3.5 `virtual bool decaf::util::logging::Level::operator< ( const Level & value ) const [virtual]`

6.475.3.6 `virtual bool decaf::util::logging::Level::operator== ( const Level & value ) const [virtual]`

6.475.3.7 `static Level decaf::util::logging::Level::parse ( const std::string & name ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]`

Parse a level name string into a **Level** (p. 2290).

The argument string may consist of either a level name or an integer value.

For example:

\* "SEVERE" \* "1000"

#### Parameters

<i>name</i>	- The name or int value of the desired <b>Level</b> (p. 2290)
-------------	---

#### Returns

the parsed **Level** (p. 2290) value, passing in a level name that is an int value that is not one of the known **Level** (p. 2290) values will result in a new **Level** (p. 2290) that has been initialized with that int value and name as the string form of the int.

#### Exceptions

<i>IllegalArgumentException</i>	if the value is not valid, validity means that the string is either a valid int (between Integer::MIN_VALUE and Integer::MAX_VALUE or is one of the known level names.
---------------------------------	--

6.475.3.8 `std::string decaf::util::logging::Level::toString ( ) const [inline]`

#### Returns

the string value of this **Level** (p. 2290), e.g. "SEVERE".

## 6.475.4 Field Documentation

### 6.475.4.1 `const Level decaf::util::logging::Level::ALL` [static]

ALL indicates that all messages should be logged.

This level is initialized to `Integer::MIN_VALUE`.

### 6.475.4.2 `const Level decaf::util::logging::Level::CONFIG` [static]

CONFIG is a message level for static configuration messages.

CONFIG messages are intended to provide a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations. For example, CONFIG message might include the CPU type, the System properties, etc. This level is initialized to 600.

### 6.475.4.3 `const Level decaf::util::logging::Level::DEBUG` [static]

DEBUG is a level for more verbose informative messages.

DEBUG messages are intended to provide a more detailed message intended for use by developers in tracking the behavior of a client. DEBUG messages typically contain more implementation specific information that might not be significant to end users or system admins. This level is initialized to 700.

### 6.475.4.4 `const Level decaf::util::logging::Level::FINE` [static]

FINE is a message level providing tracing information.

All of FINE, FINER, and FINEST are intended for relatively detailed tracing. The exact meaning of the three levels will vary between subsystems, but in general, FINEST should be used for the most detailed output, FINER for somewhat less detailed output, and FINE for the lowest volume (and most important) messages.

In general the FINE level should be used for information that will be broadly interesting to developers who do not have a specialized interest in the specific subsystem.

FINE messages might include things like minor (recoverable) failures. Issues indicating potential performance problems are also worth logging as FINE. This level is initialized to 500.

### 6.475.4.5 `const Level decaf::util::logging::Level::FINER` [static]

FINER indicates a fairly detailed tracing message.

By default logging calls for entering, returning, or throwing an exception are traced at this level. This level is initialized to 400.

#### 6.475.4.6 `const Level decaf::util::logging::Level::FINEST` [static]

FINEST indicates a highly detailed tracing message.

This level is initialized to 300.

#### 6.475.4.7 `const Level decaf::util::logging::Level::INFO` [static]

INFO is a message level for informational messages.

Typically INFO messages will be written to the console or its equivalent. So the INFO level should only be used for reasonably significant messages that will make sense to end users and system admins. This level is initialized to 800.

#### 6.475.4.8 `const Level decaf::util::logging::Level::INHERIT` [static]

NULL is a special level that indicates that the **Logger** (p. 2345) should get its **Level** (p. 2290) from its parent **Logger** (p. 2345), the value is initialized as zero.

#### 6.475.4.9 `const Level decaf::util::logging::Level::OFF` [static]

OFF is a special level that can be used to turn off logging.

This level is initialized to `Integer::MAX_VALUE`

#### 6.475.4.10 `const Level decaf::util::logging::Level::SEVERE` [static]

SEVERE is a message level indicating a serious failure.

In general SEVERE messages should describe events that are of considerable importance and which will prevent normal program execution. They should be reasonably intelligible to end users and to system administrators. This level is initialized to 1000.

#### 6.475.4.11 `const Level decaf::util::logging::Level::WARNING` [static]

WARNING is a message level indicating a potential problem.

In general WARNING messages should describe events that will be of interest to end users or system managers, or which indicate potential problems. This level is initialized to 900.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Level.h`

## 6.476 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

```
#include <src/main/decaf/util/List.h>
```

Inheritance diagram for decaf::util::List< E >:

### Public Member Functions

- **List** ()
- virtual **~List** ()
- virtual **ListIterator**< E > \* **listIterator** ()=0
- virtual **ListIterator**< E > \* **listIterator** () const =0
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual std::size\_t **indexOf** (const E &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.*
- virtual std::size\_t **lastIndexOf** (const E &value)=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.*
- virtual E **get** (std::size\_t index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Gets the element contained at position passed.*
- virtual E **set** (std::size\_t index, const E &element)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Replaces the element at the specified position in this list with the specified element.*
- virtual void **add** (std::size\_t index, const E &element)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Inserts the specified element at the specified position in this list.*
- virtual bool **addAll** (std::size\_t index, const **Collection**< E > &source)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Inserts all of the elements in the specified collection into this list at the specified position (optional operation).*
- virtual E **remove** (std::size\_t index)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Removes the element at the specified position in this list.*

### 6.476.1 Detailed Description

```
template<typename E>class decaf::util::List< E >
```

An ordered collection (also known as a sequence).

The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements *e1* and *e2* such that *e1.equals(e2)*, and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

### 6.476.2 Constructor & Destructor Documentation

6.476.2.1 `template<typename E> decaf::util::List< E >::List( ) [inline]`

6.476.2.2 `template<typename E> virtual decaf::util::List< E >::~List( ) [inline, virtual]`

### 6.476.3 Member Function Documentation

6.476.3.1 `template<typename E> virtual void decaf::util::List< E >::add ( std::size_t index, const E & element ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

#### Parameters

<i>index</i>	- index at which the specified element is to be inserted
<i>element</i>	- element to be inserted

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 3536), `decaf::util::StlList< cms::MessageConsumer * >` (p. 3536), `decaf::util::StlList< CompositeTask * >` (p. 3536), `decaf::util::StlList<`



URI > (p. 3536), decaf::util::StlList< Pointer< DestinationInfo > > (p. 3536), decaf::util::StlList< PrimitiveValueNode > (p. 3536), decaf::util::StlList< Pointer< Command > > (p. 3536), decaf::util::StlList< Pointer< BackupTransport > > (p. 3536), decaf::util::StlList< cms::MessageProducer \* > (p. 3536), decaf::util::StlList< cms::Destination \* > (p. 3536), decaf::util::StlList< cms::Session \* > (p. 3536), and decaf::util::StlList< cms::Connection \* > (p. 3536).

```
6.476.3.2 template<typename E> virtual bool decaf::util::List< E >::addAll
( std::size_t index, const Collection< E > & source ) throw (
    decaf::lang::exceptions::UnsupportedOperationException,
    decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
virtual]
```

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

#### Parameters

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The <b>Collection</b> (p. 1155) containing elements to be added to this list

#### Returns

true if this list changed as a result of the call

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implemented in decaf::util::StlList< E > (p. 3536), decaf::util::StlList< cms::MessageConsumer \* > (p. 3536), decaf::util::StlList< CompositeTask \* > (p. 3536), decaf::util::StlList< URI > (p. 3536), decaf::util::StlList< Pointer< DestinationInfo > > (p. 3536), decaf::util::StlList< PrimitiveValueNode > (p. 3536), decaf::util::StlList< Pointer< Command > > (p. 3536), decaf::util::StlList< Pointer< BackupTransport > > (p. 3536), decaf::util::StlList< cms::MessageProducer \* > (p. 3536), decaf::util::StlList< cms::Destination \* > (p. 3536), decaf::util::StlList< cms::Session \* > (p. 3536), and decaf::util::StlList< cms::Connection \* > (p. 3536).

```
6.476.3.3  template<typename E> virtual E decaf::util::List< E >::get ( std::size_t index )
           const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
           [pure virtual]
```

Gets the element contained at position passed.

#### Parameters

<i>index</i>	- position to get
--------------	-------------------

#### Returns

value at index

Implemented in **decaf::util::StlList< E >** (p. 3538), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3538), **decaf::util::StlList< CompositeTask \* >** (p. 3538), **decaf::util::StlList< URI >** (p. 3538), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3538), **decaf::util::StlList< PrimitiveValueNode >** (p. 3538), **decaf::util::StlList< Pointer< Command > >** (p. 3538), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3538), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3538), **decaf::util::StlList< cms::Destination \* >** (p. 3538), **decaf::util::StlList< cms::Session \* >** (p. 3538), and **decaf::util::StlList< cms::Connection \* >** (p. 3538).

```
6.476.3.4  template<typename E> virtual std::size_t decaf::util::List< E >::indexOf ( const
           E & value ) throw ( decaf::lang::exceptions::NoSuchElementException )
           [pure virtual]
```

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

#### Parameters

<i>value</i>	- element to search for
--------------	-------------------------

#### Returns

the index of the first occurrence of the specified element in this list,

#### Exceptions

<i>NoSuchElementException</i>	if value is not in the list
-------------------------------	-----------------------------

Implemented in **decaf::util::StlList< E >** (p. 3538), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3538), **decaf::util::StlList< CompositeTask \* >** (p. 3538), **decaf::util::StlList< URI >** (p. 3538), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3538), **decaf::util::StlList< PrimitiveValueNode >** (p. 3538), **decaf::util::StlList< Pointer< Command > >** (p. 3538), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3538),

(p. 3538), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3538), **decaf::util::StlList< cms::Destination \* >** (p. 3538), **decaf::util::StlList< cms::Session \* >** (p. 3538), and **decaf::util::StlList< cms::Connection \* >** (p. 3538).

```
6.476.3.5  template<typename E> virtual size_t decaf::util::List< E >::lastIndexOf ( const E
           & value ) throw ( decaf::lang::exceptions::NoSuchElementException )
           [pure virtual]
```

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

#### Parameters

<i>value</i>	- element to search for
--------------	-------------------------

#### Returns

the index of the last occurrence of the specified element in this list.

#### Exceptions

<i>NoSuchElementException</i>	if <i>value</i> is not in the list
-------------------------------	------------------------------------

Implemented in **decaf::util::StlList< E >** (p. 3539), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3539), **decaf::util::StlList< CompositeTask \* >** (p. 3539), **decaf::util::StlList< URI >** (p. 3539), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3539), **decaf::util::StlList< PrimitiveValueNode >** (p. 3539), **decaf::util::StlList< Pointer< Command > >** (p. 3539), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3539), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3539), **decaf::util::StlList< cms::Destination \* >** (p. 3539), **decaf::util::StlList< cms::Session \* >** (p. 3539), and **decaf::util::StlList< cms::Connection \* >** (p. 3539).

```
6.476.3.6  template<typename E> virtual ListIterator<E>* decaf::util::List< E
           >::listIterator ( ) [pure virtual]
```

#### Returns

a list iterator over the elements in this list (in proper sequence).

Implemented in **decaf::util::StlList< E >** (p. 3540), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3540), **decaf::util::StlList< CompositeTask \* >** (p. 3540), **decaf::util::StlList< URI >** (p. 3540), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3540), **decaf::util::StlList< PrimitiveValueNode >** (p. 3540), **decaf::util::StlList< Pointer< Command > >** (p. 3540), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3540), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3540), **decaf::util::StlList< cms::Destination \* >** (p. 3540), **decaf::util::StlList< cms::Session \* >** (p. 3540), and **decaf::util::StlList< cms::Connection \* >** (p. 3540).

```

6.476.3.7  template<typename E> virtual ListIterator<E>*
            decaf::util::List< E >::listIterator ( std::size_t index ) const throw (
            decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
            virtual]

```

Implemented in **decaf::util::StlList< E >** (p. 3541), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3541), **decaf::util::StlList< CompositeTask \* >** (p. 3541), **decaf::util::StlList< URI >** (p. 3541), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3541), **decaf::util::StlList< PrimitiveValueNode >** (p. 3541), **decaf::util::StlList< Pointer< Command > >** (p. 3541), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3541), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3541), **decaf::util::StlList< cms::Destination \* >** (p. 3541), **decaf::util::StlList< cms::Session \* >** (p. 3541), and **decaf::util::StlList< cms::Connection \* >** (p. 3541).

```

6.476.3.8  template<typename E> virtual ListIterator<E>* decaf::util::List< E
            >::listIterator ( ) const [pure virtual]

```

Implemented in **decaf::util::StlList< E >** (p. 3540), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3540), **decaf::util::StlList< CompositeTask \* >** (p. 3540), **decaf::util::StlList< URI >** (p. 3540), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3540), **decaf::util::StlList< PrimitiveValueNode >** (p. 3540), **decaf::util::StlList< Pointer< Command > >** (p. 3540), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3540), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3540), **decaf::util::StlList< cms::Destination \* >** (p. 3540), **decaf::util::StlList< cms::Session \* >** (p. 3540), and **decaf::util::StlList< cms::Connection \* >** (p. 3540).

```

6.476.3.9  template<typename E> virtual ListIterator<E>*
            decaf::util::List< E >::listIterator ( std::size_t index ) throw (
            decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
            virtual]

```

#### Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

#### Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if the index is out of range (index < 0    index > <b>size()</b> (p. 1164))
----------------------------------	---

Implemented in **decaf::util::StlList< E >** (p. 3540), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3540), **decaf::util::StlList< CompositeTask \* >** (p. 3540), **decaf::util::StlList<**

**URI** > (p. 3540), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3540), **decaf::util::StlList< PrimitiveValueNode >** (p. 3540), **decaf::util::StlList< Pointer< Command > >** (p. 3540), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3540), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3540), **decaf::util::StlList< cms::Destination \* >** (p. 3540), **decaf::util::StlList< cms::Session \* >** (p. 3540), and **decaf::util::StlList< cms::Connection \* >** (p. 3540).

```
6.476.3.10  template<typename E> virtual E decaf::util::List< E >::remove ( std::size_t in-
            dex ) throw ( decaf::lang::exceptions::UnsupportedOperationException,
            decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
            virtual]
```

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

#### Parameters

<i>index</i>	- the index of the element to be removed
--------------	--

#### Returns

the element previously at the specified position

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 3541), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3541), **decaf::util::StlList< CompositeTask \* >** (p. 3541), **decaf::util::StlList< URI >** (p. 3541), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3541), **decaf::util::StlList< PrimitiveValueNode >** (p. 3541), **decaf::util::StlList< Pointer< Command > >** (p. 3541), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3541), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3541), **decaf::util::StlList< cms::Destination \* >** (p. 3541), **decaf::util::StlList< cms::Session \* >** (p. 3541), and **decaf::util::StlList< cms::Connection \* >** (p. 3541).

```
6.476.3.11  template<typename E> virtual E decaf::util::List< E
            >::set ( std::size_t index, const E & element ) throw (
            decaf::lang::exceptions::IndexOutOfBoundsException ) [pure
            virtual]
```

Replaces the element at the specified position in this list with the specified element.

#### Parameters

<i>index</i>	- index of the element to replace
<i>element</i>	- element to be stored at the specified position

**Returns**

the element previously at the specified position

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
----------------------------------	-------------------------------------

Implemented in **decaf::util::StlList< E >** (p. 3542), **decaf::util::StlList< cms::MessageConsumer \* >** (p. 3542), **decaf::util::StlList< CompositeTask \* >** (p. 3542), **decaf::util::StlList< URI >** (p. 3542), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 3542), **decaf::util::StlList< PrimitiveValueNode >** (p. 3542), **decaf::util::StlList< Pointer< Command > >** (p. 3542), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 3542), **decaf::util::StlList< cms::MessageProducer \* >** (p. 3542), **decaf::util::StlList< cms::Destination \* >** (p. 3542), **decaf::util::StlList< cms::Session \* >** (p. 3542), and **decaf::util::StlList< cms::Connection \* >** (p. 3542).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**List.h**

## 6.477 decaf::util::ListIterator< E > Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

```
#include <src/main/decaf/util/ListIterator.h>
```

Inheritance diagram for decaf::util::ListIterator< E >:

**Public Member Functions**

- virtual **~ListIterator** ()
- virtual void **add** (const E &e)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the list (optional operation).*
- virtual void **set** (const E &e)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Replaces the last element returned by next or previous with the specified element (optional operation).*

- virtual bool **hasPrevious** () const =0  
*Returns true if this list iterator has more elements when traversing the list in the reverse direction.*
- virtual E **previous** ()=0  
*Returns the previous element in the list.*
- virtual int **nextIndex** () const =0  
*Returns the index of the element that would be returned by a subsequent call to next.*
- virtual int **previousIndex** () const =0  
*Returns the index of the element that would be returned by a subsequent call to previous.*

### 6.477.1 Detailed Description

template<typename E>class decaf::util::ListIterator< E >

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Note that the **remove()** (p. 2115) and **set(Object)** methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 2115) or **previous()** (p. 2305).

### 6.477.2 Constructor & Destructor Documentation

6.477.2.1 template<typename E > virtual decaf::util::ListIterator< E >::~ListIterator ( ) [inline, virtual]

### 6.477.3 Member Function Documentation

6.477.3.1 template<typename E > virtual void decaf::util::ListIterator< E >::add ( const E & e ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException ) [pure virtual]

Inserts the specified element into the list (optional operation).

The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

#### Parameters

<i>e</i>	- the element to insert.
----------	--------------------------

**Exceptions**

<i>UnsupportedOperationException</i>	- if the add method is not supported by this list iterator.
<i>IllegalArgumentException</i>	- if some aspect of this element prevents it from being added to this list.

**6.477.3.2** `template<typename E> virtual bool decaf::util::ListIterator< E >::hasPrevious ( ) const [pure virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction.

(In other words, returns true if previous would return an element rather than throwing an exception.)

**Returns**

true if the list iterator has more elements when traversing the list in the reverse direction.

**6.477.3.3** `template<typename E> virtual int decaf::util::ListIterator< E >::nextIndex ( ) const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to next.

(Returns list size if the list iterator is at the end of the list.)

**Returns**

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

**6.477.3.4** `template<typename E> virtual E decaf::util::ListIterator< E >::previous ( ) [pure virtual]`

Returns the previous element in the list.

This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

**Returns**

the previous element in the list.

**Exceptions**

<i>NoSuchElementException</i>	- if the iteration has no previous element.
-------------------------------	---



6.477.3.5 `template<typename E> virtual int decaf::util::ListIterator< E >::previousIndex ( ) const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to previous.

(Returns -1 if the list iterator is at the beginning of the list.)

### Returns

the index of the element that would be returned by a subsequent call to previous, or -1 if list iterator is at beginning of list.

6.477.3.6 `template<typename E> virtual void decaf::util::ListIterator< E >::set ( const E & e ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException ) [pure virtual]`

Replaces the last element returned by next or previous with the specified element (optional operation).

This call can be made only if neither **ListIterator.remove** (p. 2115) nor **ListIterator.add** (p. 2304) have been called after the last call to next or previous.

### Parameters

<i>e</i>	- the element with which to replace the last element returned by next or previous.
----------	--

### Exceptions

<i>UnsupportedOperationException</i>	- if the add method is not supported by this list iterator.
<i>IllegalArgumentException</i>	- if some aspect of this element prevents it from being added to this list.
<i>IllegalStateException</i>	- if neither next nor previous have been called, or remove or add have been called after the last call to next or previous.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ListIterator.h`

## 6.478 activemq::commands::LocalTransactionId Class Reference

```
#include <src/main/activemq/commands/LocalTransactionId.h>
```

Inheritance diagram for activemq::commands::LocalTransactionId:

## Public Types

- typedef **decaf::lang::PointerComparator**< **LocalTransactionId** > **COMPARATOR**

## Public Member Functions

- **LocalTransactionId** ()
- **LocalTransactionId** (const **LocalTransactionId** &other)
- virtual ~**LocalTransactionId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **LocalTransactionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual int **compareTo** (const **LocalTransactionId** &value) const
- virtual bool **equals** (const **LocalTransactionId** &value) const
- virtual bool **operator==** (const **LocalTransactionId** &value) const
- virtual bool **operator<** (const **LocalTransactionId** &value) const
- **LocalTransactionId** & **operator=** (const **LocalTransactionId** &other)

## Static Public Attributes

- static const unsigned char **ID\_LOCALTRANSACTIONID** = 111

## Protected Attributes

- long long **value**
- **Pointer**< **ConnectionId** > **connectionId**

### 6.478.1 Member Typedef Documentation

6.478.1.1 `typedef decaf::lang::PointerComparator<LocalTransactionId>  
activemq::commands::LocalTransactionId::COMPARATOR`

Reimplemented from **activemq::commands::TransactionId** (p. 3760).

### 6.478.2 Constructor & Destructor Documentation

6.478.2.1 `activemq::commands::LocalTransactionId::LocalTransactionId ( )`

6.478.2.2 `activemq::commands::LocalTransactionId::LocalTransactionId ( const  
LocalTransactionId & other )`

6.478.2.3 `virtual activemq::commands::LocalTransactionId::~~LocalTransactionId ( )  
[virtual]`

### 6.478.3 Member Function Documentation

6.478.3.1 `virtual LocalTransactionId* ac-  
tivemq::commands::LocalTransactionId::cloneDataStructure ( )  
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Reimplemented from **activemq::commands::TransactionId** (p. 3761).

6.478.3.2 `virtual int activemq::commands::LocalTransactionId::compareTo ( const  
LocalTransactionId & value ) const [virtual]`

6.478.3.3 `virtual void activemq::commands::LocalTransactionId::copyDataStructure ( const  
DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::TransactionId** (p. 3761).

6.478.3.4 `virtual bool activemq::commands::LocalTransactionId::equals ( const  
DataSet * value ) const` [virtual]

Compares the **DataSet** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataSet's are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 3761).

6.478.3.5 `virtual bool activemq::commands::LocalTransactionId::equals ( const  
LocalTransactionId & value ) const` [virtual]

6.478.3.6 `virtual const Pointer<ConnectionId>& ac-  
 tivemq::commands::LocalTransactionId::getConnectionId ( ) const`  
 [virtual]

6.478.3.7 `virtual Pointer<ConnectionId>& ac-  
 tivemq::commands::LocalTransactionId::getConnectionId ( )`  
 [virtual]

6.478.3.8 `virtual unsigned char activemq::commands::LocalTransactionId::getDataSetType  
 ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSet** (p. 1628) type copy.

Reimplemented from **activemq::commands::TransactionId** (p. 3762).

6.478.3.9 `virtual long long activemq::commands::LocalTransactionId::getValue ( ) const`  
 [virtual]

6.478.3.10 `virtual bool activemq::commands::LocalTransactionId::operator< ( const  
LocalTransactionId & value ) const` [virtual]

6.478.3.11 `LocalTransactionId& activemq::commands::LocalTransactionId::operator= (`  
`const LocalTransactionId & other )`

6.478.3.12 `virtual bool activemq::commands::LocalTransactionId::operator== ( const  
LocalTransactionId & value ) const` [virtual]

6.479

**activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**

**Class Reference**

2319

6.478.3.13 virtual void activemq::commands::LocalTransactionId::setConnectionId ( const  
Pointer< ConnectionId > & connectionId ) [virtual]

6.478.3.14 virtual void activemq::commands::LocalTransactionId::setValue ( long long value )  
[virtual]

6.478.3.15 virtual std::string activemq::commands::LocalTransactionId::toString ( ) const  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 3762).

#### 6.478.4 Field Documentation

6.478.4.1 Pointer<ConnectionId> activemq::commands::LocalTransactionId::connectionId  
[protected]

6.478.4.2 const unsigned char activemq::commands::LocalTransactionId::ID\_  
LOCALTRANSACTIONID = 111 [static]

6.478.4.3 long long activemq::commands::LocalTransactionId::value  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**LocalTransactionId.h**

## 6.479 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2310).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarsha
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller:

#### Public Member Functions

- **LocalTransactionIdMarshaller** ()

- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.479.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2310).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.479.2 Constructor & Destructor Documentation

6.479.2.1 **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller**  
 ( ) [inline]

6.479.2.2 **virtual activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller**  
 ( ) [inline, virtual]

### 6.479.3 Member Function Documentation

6.479.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

6.479

**activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**

**Class Reference**

**2321**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.479.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.479.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3782).

6.479.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3783).

```
6.479.3.5  virtual int activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3783).

```
6.479.3.6  virtual void activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



6.480

**activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

**Class Reference**

**2323**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3784).

6.479.3.7 virtual void **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**LocalTransactionIdMarshaller.h**

## 6.480 **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2314).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**:

#### Public Member Functions

- **LocalTransactionIdMarshaller** ()

- virtual `~LocalTransactionIdMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.480.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2314).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.480.2 Constructor & Destructor Documentation

6.480.2.1 `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller ( ) [inline]`

6.480.2.2 `virtual activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller ( ) [inline, virtual]`

### 6.480.3 Member Function Documentation

6.480.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

6.480

**activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

**Class Reference**

2325

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.480.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.480.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3771).

6.480.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3772).

```
6.480.3.5  virtual int activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3772).

```
6.480.3.6  virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.481

**activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

**Class Reference**

**2327**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3773).

6.480.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3773).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LocalTransactionIdMarshaller.h**

## 6.481 **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2318).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**:

#### Public Member Functions

- **LocalTransactionIdMarshaller** ()

- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.481.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2318).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.481.2 Constructor & Destructor Documentation

6.481.2.1 **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller**  
 ( ) [inline]

6.481.2.2 **virtual activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller**  
 ( ) [inline, virtual]

### 6.481.3 Member Function Documentation

6.481.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

6.481

**activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

**Class Reference**

2329

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.481.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.481.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3775).

6.481.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3775).

```
6.481.3.5  virtual int activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3776).

```
6.481.3.6  virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



6.482

**activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

**Class Reference**

**2331**

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3776).

6.481.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightUnmarshal** ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**LocalTransactionIdMarshaller.h**

## 6.482 **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2322).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**:

#### Public Member Functions

- **LocalTransactionIdMarshaller** ()

- virtual `~LocalTransactionIdMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.482.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2322).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.482.2 Constructor & Destructor Documentation

6.482.2.1 `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller ( ) [inline]`

6.482.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller ( ) [inline, virtual]`

### 6.482.3 Member Function Documentation

6.482.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

6.482

**activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

**Class Reference**

**2333**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.482.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.482.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3764).

6.482.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3764).

```
6.482.3.5  virtual int activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3765).

```
6.482.3.6  virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

6.483

**activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

**Class Reference**

2335

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3765).

6.482.3.7 virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3766).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**LocalTransactionIdMarshaller.h**

## 6.483 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2326).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**:

#### Public Member Functions

- **LocalTransactionIdMarshaller** ()

- virtual `~LocalTransactionIdMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.483.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2326).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.483.2 Constructor & Destructor Documentation

6.483.2.1 `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller ( ) [inline]`

6.483.2.2 `virtual activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller ( ) [inline, virtual]`

### 6.483.3 Member Function Documentation

6.483.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

6.483

**activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

**Class Reference**

**2337**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.483.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.483.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3779).

6.483.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3779).

```
6.483.3.5  virtual int activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3780).

```
6.483.3.6  virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



6.484

**activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

**Class Reference**

2339

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3780).

6.483.3.7 virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3781).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LocalTransactionIdMarshaller.h**

## 6.484 **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2330).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarsha
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**:

#### Public Member Functions

- **LocalTransactionIdMarshaller** ()

- virtual `~LocalTransactionIdMarshaller ()`
- virtual `commands::DataStructure * createObject () const`  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType () const`  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw ( decaf::io::IOException )`  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw ( decaf::io::IOException )`  
*Write a object instance to data output stream.*

### 6.484.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2330).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.484.2 Constructor & Destructor Documentation

6.484.2.1 `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller ( ) [inline]`

6.484.2.2 `virtual activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller ( ) [inline, virtual]`

### 6.484.3 Member Function Documentation

6.484.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

6.484

**activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

**Class Reference**

**2341**

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.484.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.484.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3767).

6.484.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3768).

```
6.484.3.5  virtual int activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3768).

```
6.484.3.6  virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3769).

6.484.3.7 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**LocalTransactionIdMarshaller.h**

**6.485 decaf::util::concurrent::Lock Class Reference**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

```
#include <src/main/decaf/util/concurrent/Lock.h>
```

**Public Member Functions**

- **Lock (Synchronizable** \*object, const bool initiallyLocked=true)  
*Constructor - initializes the object member and locks the object if desired.*
- virtual ~**Lock** ()  
*Destructor - Unlocks the object if it is locked.*
- void **lock** ()

*Locks the object.*

- void **unlock** ()

*Unlocks the object if it is already locked, otherwise a call to this method has no effect.*

- bool **isLocked** () const

*Indicates whether or not the object is locked.*

### 6.485.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

#### Since

1.0

### 6.485.2 Constructor & Destructor Documentation

6.485.2.1 `decaf::util::concurrent::Lock ( Synchronizable * object, const bool initiallyLocked = true )`

Constructor - initializes the object member and locks the object if desired.

#### Parameters

<i>object</i>	The sync object to control
<i>initially-Locked</i>	If true, the object will automatically be locked.

6.485.2.2 `virtual decaf::util::concurrent::Lock::~~Lock ( ) [virtual]`

Destructor - Unlocks the object if it is locked.

### 6.485.3 Member Function Documentation

6.485.3.1 `bool decaf::util::concurrent::Lock::isLocked ( ) const [inline]`

Indicates whether or not the object is locked.

#### Returns

true if the object is locked, otherwise false.

6.485.3.2 `void decaf::util::concurrent::Lock::lock ( )`

Locks the object.

## 6.485.3.3 void decaf::util::concurrent::Lock::unlock ( )

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Lock.h**

## 6.486 decaf::util::concurrent::locks::Lock Class Reference

**Lock** (p. 2336) implementations provide more extensive locking operations than can be obtained using synchronized statements.

```
#include <src/main/decaf/util/concurrent/locks/Lock.h>
```

Inheritance diagram for decaf::util::concurrent::locks::Lock:

## Public Member Functions

- virtual **~Lock** ( )
- virtual void **lock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock.*
- virtual void **lockInterruptibly** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock unless the current thread is interrupted.*
- virtual bool **tryLock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock only if it is free at the time of invocation.*
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.*
- virtual void **unlock** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Releases the lock.*
- virtual **Condition** \* **newCondition** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns a new **Condition** (p. 1220) instance that is bound to this **Lock** (p. 2336) instance.*

## 6.486.1 Detailed Description

**Lock** (p. 2336) implementations provide more extensive locking operations than can be obtained using synchronized statements.

They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 1220) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. Commonly, a lock provides exclusive access to a shared resource: only one thread at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some locks may allow concurrent access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 3117).

While the scoping mechanism for synchronized statements makes it much easier to program with monitor locks, and helps avoid many common programming errors involving locks, there are occasions where you need to work with locks in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock** (p. 2336) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple locks to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of locks that occurs with synchronized statements. In most cases, the following idiom should be used:

```
Lock (p. 2336) l = ...; l.lock(); try { // access the resource protected by this lock } catch(...)
{ l.unlock(); }
```

When locking and unlocking occur in different scopes, care must be taken to ensure that all code that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

**Lock** (p. 2336) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 2340)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 2338)), and an attempt to acquire the lock that can timeout (**tryLock(long, TimeUnit)**).

Note that **Lock** (p. 2336) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 2336) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.



**Since**

1.0

**6.486.2 Constructor & Destructor Documentation**

6.486.2.1 virtual decaf::util::concurrent::locks::Lock::~~Lock ( ) [inline, virtual]

**6.486.3 Member Function Documentation**

6.486.3.1 virtual void decaf::util::concurrent::locks::Lock::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Acquires the lock.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

Implementation Considerations

A **Lock** (p. 2336) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2336) implementation.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3130).

6.486.3.2 virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException ) [pure virtual]

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p. 2336) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2336) implementation.

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3130).

```
6.486.3.3 virtual Condition* decaf::util::concurrent::locks::Lock::newCondition
( ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::UnsupportedOperationException ) [pure
virtual]
```

Returns a new **Condition** (p. 1220) instance that is bound to this **Lock** (p. 2336) instance.

Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()** (p. 1222) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

### Implementation Considerations

The exact operation of the **Condition** (p. 1220) instance depends on the **Lock** (p. 2336) implementation and must be documented by that implementation.

### Returns

A new **Condition** (p. 1220) instance for this **Lock** (p. 2336) instance the caller must delete the returned **Condition** (p. 1220) object when done with it.

### Exceptions

<i>RuntimeException</i>	if an error occurs while creating the <b>Condition</b> (p. 1220).
<i>UnsupportedOperationException</i>	if this <b>Lock</b> (p. 2336) implementation does not support conditions

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3131).

6.486.3.4 `virtual bool decaf::util::concurrent::locks::Lock::tryLock ( long long time, const TimeUnit & unit ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.

If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- \* The lock is acquired by the current thread; or
- \* Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or
- \* The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

- \* has its interrupted status set on entry to this method; or
- \* is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p. 2336) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 2336) implementation.

#### Parameters

<i>time</i>	the maximum time to wait for the lock
<i>unit</i>	the time unit of the time argument

#### Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

#### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3131).

6.486.3.5 `virtual bool decaf::util::concurrent::locks::Lock::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]`

Acquires the lock only if it is free at the time of invocation.

Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

**Lock** (p. 2336) `lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }`

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

#### Returns

true if the lock was acquired and false otherwise

#### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3133).

6.486.3.6 `virtual void decaf::util::concurrent::locks::Lock::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]`

Releases the lock.

#### Implementation Considerations

A **Lock** (p. 2336) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 2336) implementation.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>IllegalMonitorStateException</i>	if the current thread is not the owner of the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 3133).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/Lock.h`

## 6.487 decaf::util::concurrent::locks::LockSupport Class Reference

Basic thread blocking primitives for creating locks and other synchronization classes.

```
#include <src/main/decaf/util/concurrent/locks/LockSupport.h>
```

### Public Member Functions

- `~LockSupport ()`

### Static Public Member Functions

- static void **unpark** (`decaf::lang::Thread *thread`) throw ()  
*Makes available the permit for the given thread, if it was not already available.*
- static void **park** () throw ()  
*Disables the current thread for thread scheduling purposes unless the permit is available.*
- static void **parkNanos** (long long nanos) throw ()  
*Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.*
- static void **parkUntil** (long long deadline) throw ()  
*Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.*

### 6.487.1 Detailed Description

Basic thread blocking primitives for creating locks and other synchronization classes.

This class associates, with each thread that uses it, a permit (in the sense of the **Semaphore** (p. 3280) class). A call to `park` will return immediately if the permit is available, consuming it in the process; otherwise it may block. A call to `unpark` makes the permit available, if it was not already available. (Unlike with Semaphores though, permits do not accumulate. There is at most one.)

Methods `park` and `unpark` provide efficient means of blocking and unblocking threads. Races between one thread invoking `park` and another thread trying to `unpark` it will preserve liveness, due to the permit. Additionally, `park` will return if the caller's thread was interrupted, and timeout versions are supported. The `park` method may also return at any other time, for "no reason", so in general must be invoked within a loop that rechecks conditions upon return. In this sense `park` serves as an optimization of a "busy wait" that does not waste as much time spinning, but must be paired with an `unpark` to be effective.

These methods are designed to be used as tools for creating higher-level synchronization utilities, and are not in themselves useful for most concurrency control applications. The park method is designed for use only in constructions of the form:

```
while (!canProceed()) { ... LockSupport.park(this); }
```

where neither canProceed nor any other actions prior to the call to park entail locking or blocking. Because only one permit is associated with each thread, any intermediary uses of park could interfere with its intended effects.

Sample Usage. Here is a sketch of a first-in-first-out non-reentrant lock class:

```
class FIFOMutex { private:
    AtomicBoolean locked; ConcurrentLinkedQueue<Thread*> waiters;
public:
    void lock() {
        bool wasInterrupted = false; Thread* current = Thread::currentThread(); waiters.add(
            current );
        // Block while not first in queue or cannot acquire lock while( waiters.peek() != current ||
        !locked.compareAndSet( false, true ) ) {
            LockSupport.park(this); if( Thread::interrupted() ) // ignore interrupts while waiting was-
            Interrupted = true; }
        waiters.remove(); if( wasInterrupted ) // reassert interrupt status on exit current.interrupt();
    }
    void unlock() { locked.set( false ); LockSupport.unpark (p. 2344)( waiters.peek() ); } }
```

#### Since

1.0

### 6.487.2 Constructor & Destructor Documentation

6.487.2.1 `decaf::util::concurrent::locks::LockSupport::~~LockSupport ( )`

### 6.487.3 Member Function Documentation

6.487.3.1 `static void decaf::util::concurrent::locks::LockSupport::park ( ) throw ()`  
`[static]`

Disables the current thread for thread scheduling purposes unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes unpark with the current thread as the target; or \* Some other thread interrupts the current thread; or \* The call spuriously (that is, for no reason)

returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread upon return.

**6.487.3.2** `static void decaf::util::concurrent::locks::LockSupport::parkNanos ( long long nanos ) throw () [static]`

Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

\* Some other thread invokes `unpark` with the current thread as the target; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses; or \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the elapsed time upon return.

#### Parameters

<i>nanos</i>	the maximum number of nanoseconds to wait
--------------	---

**6.487.3.3** `static void decaf::util::concurrent::locks::LockSupport::parkUntil ( long long deadline ) throw () [static]`

Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

\* Some other thread invokes `unpark` with the current thread as the target; or \* Some other thread interrupts the current thread; or \* The specified deadline passes; or \* The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the current time upon return.

#### Parameters

<i>deadline</i>	the absolute time, in milliseconds from the Epoch, to wait until
-----------------	--

6.487.3.4 `static void decaf::util::concurrent::locks::LockSupport::unpark ( decaf::lang::Thread * thread ) throw () [static]`

Makes available the permit for the given thread, if it was not already available.

If the thread was blocked on park then it will unblock. Otherwise, its next call to park is guaranteed not to block. This operation is not guaranteed to have any effect at all if the given thread has not been started.

#### Parameters

<code>thread</code>	the thread to unport, or NULL in which case the method has no effect.
---------------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/LockSupport.h`

## 6.488 decaf::util::logging::Logger Class Reference

A **Logger** (p. 2345) object is used to log messages for a specific system or application component.

```
#include <src/main/decaf/util/logging/Logger.h>
```

#### Public Member Functions

- virtual `~Logger ()`
- `const std::string & getName () const`  
*Gets the name of this **Logger** (p. 2345).*
- `Logger * getParent () const`  
*Gets the parent of this **Logger** (p. 2345) which will be the nearest existing **Logger** (p. 2345) in this Loggers namespace.*
- `void setParent (Logger *parent)`  
*Set (p. 3379) the parent for this **Logger** (p. 2345).*
- `void addHandler (Handler *handler) throw ( lang::exceptions::NullPointerException )`  
*Add a log **Handler** (p. 1941) to receive logging messages.*
- `void removeHandler (Handler *handler)`  
*Removes the specified **Handler** (p. 1941) from this logger, ownership of the **Handler** (p. 1941) pointer is returned to the caller.*
- `const std::list< Handler * > & getHandlers () const`  
*Gets a vector containing all the handlers that this class has been assigned to use.*
- `void setFilter (Filter *filter)`  
*Set (p. 3379) a filter to control output on this **Logger** (p. 2345).*
- `const Filter * getFilter () const`  
*Gets the **Filter** (p. 1853) object that this class is using.*



- **Level** **getLevel** () const  
*Get the log **Level** (p. 2290) that has been specified for this **Logger** (p. 2345).*
- void **setLevel** (const **Level** &level)  
***Set** (p. 3379) the log level specifying which message levels will be logged by this logger.*
- bool **getUseParentHandlers** () const  
*Discover whether or not this logger is sending its output to its parent logger.*
- void **setUseParentHandlers** (bool value)  
*Specify whether or not this logger should send its output to it's parent **Logger** (p. 2345).*
- virtual void **entering** (const std::string &blockName, const std::string &file, const int line)  
*Logs an Block Enter message.*
- virtual void **exiting** (const std::string &blockName, const std::string &file, const int line)  
*Logs an Block Exit message.*
- virtual void **severe** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a SEVERE **Level** (p. 2290) Log.*
- virtual void **warning** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a WARN **Level** (p. 2290) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a INFO **Level** (p. 2290) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a DEBUG **Level** (p. 2290) Log.*
- virtual void **config** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a CONFIG **Level** (p. 2290) Log.*
- virtual void **fine** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a FINE **Level** (p. 2290) Log.*
- virtual void **finer** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a FINER **Level** (p. 2290) Log.*
- virtual void **finest** (const std::string &file, const int line, const std::string functionName, const std::string &message)  
*Log a FINEST **Level** (p. 2290) Log.*
- virtual void **throwing** (const std::string &file, const int line, const std::string functionName, const decaf::lang::Throwable &thrown)  
*Log throwing an exception.*
- virtual bool **isLoggable** (const **Level** &level) const  
*Check if a message of the given level would actually be logged by this logger.*
- virtual void **log** (**LogRecord** &record)

Log a **LogRecord** (p. 2370).

- virtual void **log** (const **Level** &level, const std::string &message)

Log a message, with no arguments.

- virtual void **log** (const **Level** &levels, const std::string &file, const int line, const std::string &message,...)

Log a message, with the list of params that is formatted into the message string.

- virtual void **log** (const **Level** &level, const std::string &file, const int line, const std::string &message, **lang::Exception** &ex)

Log a message, with associated Throwable information.

### Static Public Member Functions

- static **Logger** \* **getAnonymousLogger** ()

Creates an anonymous logger.

- static **Logger** \* **getLogger** (const std::string &name)

Find or create a logger for a named subsystem.

### Protected Member Functions

- **Logger** (const std::string &name)

Creates a new instance of the **Logger** (p. 2345) with the given name.

#### 6.488.1 Detailed Description

A **Logger** (p. 2345) object is used to log messages for a specific system or application component.

Loggers are normally named, using a hierarchical dot-separated namespace. **Logger** (p. 2345) names can be arbitrary strings, but they should normally be based on the namespace or class name of the logged component, such as **decaf.net** (p. 134) or **org.apache.decaf**. In addition it is possible to create "anonymous" Loggers that are not stored in the **Logger** (p. 2345) namespace.

**Logger** (p. 2345) objects may be obtained by calls on one of the **getLogger** factory methods. These will either create a new **Logger** (p. 2345) or return a suitable existing **Logger** (p. 2345).

Logging messages will be forwarded to registered **Handler** (p. 1941) objects, which can forward the messages to a variety of destinations, including consoles, files, OS logs, etc.

Each **Logger** (p. 2345) keeps track of a "parent" **Logger** (p. 2345), which is its nearest existing ancestor in the **Logger** (p. 2345) namespace.

Each **Logger** (p. 2345) has a "Level" associated with it. This reflects a minimum **Level** (p. 2290) that this logger cares about. If a Logger's level is set to **Level::INHERIT** (p. 2295), then its effective level is inherited from its parent, which may in turn obtain it recursively from its parent, and so on up the tree.

The log level can be configured based on the properties from the logging configuration file, as described in the description of the **LogManager** (p. 2363) class. However it may also be dynamically changed by calls on the **Logger.setLevel** (p. 2355) method. If a logger's level is changed the change may also affect child loggers, since any child logger that has 'inherit' as its level will inherit its effective level from its parent.

On each logging call the **Logger** (p. 2345) initially performs a cheap check of the request level (e.g. SEVERE or FINE) against the effective log level of the logger. If the request level is lower than the log level, the logging call returns immediately.

After passing this initial (cheap) test, the **Logger** (p. 2345) will allocate a **LogRecord** (p. 2370) to describe the logging message. It will then call a **Filter** (p. 1853) (if present) to do a more detailed check on whether the record should be published. If that passes it will then publish the **LogRecord** (p. 2370) to its output Handlers. By default, loggers also publish to their parent's Handlers, recursively up the tree.

Formatting is the responsibility of the output **Handler** (p. 1941), which will typically call a **Formatter** (p. 1927).

Note that formatting need not occur synchronously. It may be delayed until a **LogRecord** (p. 2370) is actually written to an external sink.

All methods on **Logger** (p. 2345) are thread safe.

#### Since

1.0

### 6.488.2 Constructor & Destructor Documentation

#### 6.488.2.1 decaf::util::logging::Logger::Logger ( const std::string & name ) [protected]

Creates a new instance of the **Logger** (p. 2345) with the given name.

The logger will be initially configured with a null **Level** (p. 2290) and with useParentHandlers true.

#### Parameters

<i>name</i>	A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as <b>decaf.net</b> (p. 134) or org.apache.decaf. It may be empty for anonymous Loggers.
-------------	---

#### 6.488.2.2 virtual decaf::util::logging::Logger::~Logger ( ) [virtual]

### 6.488.3 Member Function Documentation

6.488.3.1 `void decaf::util::logging::Logger::addHandler ( Handler * handler ) throw ( lang::exceptions::NullPointerException )`

Add a log **Handler** (p. 1941) to receive logging messages.

By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p. 2345) is configured with a set of Handlers that essentially act as default handlers for all loggers.

The ownership of the given **Handler** (p. 1941) is passed to the **Logger** (p. 2345) and the **Handler** (p. 1941) will be deleted when this **Logger** (p. 2345) is destroyed unless the caller first calls `removeHandler` with the same pointer value as was originally given.

#### Parameters

<i>handler</i>	A Logging <b>Handler</b> (p. 1941)
----------------	------------------------------------

#### Exceptions

<i>NullPointerException</i>	if the <b>Handler</b> (p. 1941) given is NULL.
-----------------------------	--

6.488.3.2 `virtual void decaf::util::logging::Logger::config ( const std::string & file, const int line, const std::string functionName, const std::string & message ) [virtual]`

Log a CONFIG **Level** (p. 2290) Log.

If the logger is currently enabled for the CONFIG message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2290).

6.488.3.3 `virtual void decaf::util::logging::Logger::debug ( const std::string & file, const int line, const std::string functionName, const std::string & message ) [virtual]`

Log a DEBUG **Level** (p. 2290) Log.

If the logger is currently enabled for the DEBUG message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.

<i>message</i>	The message to log at this <b>Level</b> (p. 2290).
----------------	--

6.488.3.4 virtual void decaf::util::logging::Logger::entering ( const std::string & *blockName*,  
const std::string & *file*, const int *line* ) [virtual]

Logs an Block Enter message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2294) log level.

#### Parameters

<i>blockName</i>	The source block name, (usually <code>ClassName::MethodName</code> , or <code>Method-Name</code> ).
<i>file</i>	The source file name where this method was called from.
<i>line</i>	The source line number where this method was called from.

6.488.3.5 virtual void decaf::util::logging::Logger::exiting ( const std::string & *blockName*,  
const std::string & *file*, const int *line* ) [virtual]

Logs an Block Exit message.

This is a convenience method that is used to tag a block enter, a log record with the given information is logged at the **Level::FINER** (p. 2294) log level.

#### Parameters

<i>blockName</i>	The source block name, (usually <code>ClassName::MethodName</code> , or <code>Method-Name</code> ).
<i>file</i>	The source file name where this method was called from.
<i>line</i>	The source line number where this method was called from.

6.488.3.6 virtual void decaf::util::logging::Logger::fine ( const std::string & *file*, const int *line*,  
const std::string *functionName*, const std::string & *message* ) [virtual]

Log a FINE **Level** (p. 2290) Log.

If the logger is currently enabled for the FINE message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2290).

6.488.3.7 `virtual void decaf::util::logging::Logger::finer ( const std::string & file, const int line, const std::string functionName, const std::string & message )` [virtual]

Log a **FINER Level** (p. 2290) Log.

If the logger is currently enabled for the **FINER** message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2290).

6.488.3.8 `virtual void decaf::util::logging::Logger::finest ( const std::string & file, const int line, const std::string functionName, const std::string & message )` [virtual]

Log a **FINEST Level** (p. 2290) Log.

If the logger is currently enabled for the **FINEST** message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2290).

6.488.3.9 `static Logger* decaf::util::logging::Logger::getAnonymousLogger ( )`  
[static]

Creates an anonymous logger.

The newly created **Logger** (p. 2345) is not registered in the **LogManager** (p. 2363) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger ("" ) as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

#### Returns

Newly created anonymous logger

6.488.3.10 `const Filter* decaf::util::logging::Logger::getFilter ( ) const` `[inline]`

Gets the **Filter** (p. 1853) object that this class is using.

#### Returns

the **Filter** (p. 1853) in use, (can be NULL).

6.488.3.11 `const std::list<Handler*>& decaf::util::logging::Logger::getHandlers ( ) const`

Gets a vector containing all the handlers that this class has been assigned to use.

#### Returns

a list of handlers that are used by this logger

6.488.3.12 `Level decaf::util::logging::Logger::getLevel ( ) const` `[inline]`

Get the log **Level** (p. 2290) that has been specified for this **Logger** (p. 2345).

The result may be the INHERIT level, which means that this logger's effective level will be inherited from its parent.

#### Returns

the level that is currently set

6.488.3.13 `static Logger* decaf::util::logging::Logger::getLogger ( const std::string & name )`  
`[static]`

Find or create a logger for a named subsystem.

If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 2363) and it will be configured to also send logging output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 2363) global namespace.

#### Parameters

<i>name</i>	- A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as cms or <b>activemq.core.ActiveMQConnection</b> (p. 244)
-------------	---

#### Returns

a suitable logger.

6.488.3.14 `const std::string& decaf::util::logging::Logger::getName ( ) const [inline]`

Gets the name of this **Logger** (p. 2345).

#### Returns

logger name

6.488.3.15 `Logger* decaf::util::logging::Logger::getParent ( ) const [inline]`

Gets the parent of this **Logger** (p. 2345) which will be the nearest existing **Logger** (p. 2345) in this Loggers namespace.

If this is the Root **Logger** (p. 2345) than this method returns NULL.

#### Returns

Pointer to this Loggers nearest parent **Logger** (p. 2345).

6.488.3.16 `bool decaf::util::logging::Logger::getUseParentHandlers ( ) const [inline]`

Discover whether or not this logger is sending its output to its parent logger.

#### Returns

true if using Parent Handlers

6.488.3.17 `virtual void decaf::util::logging::Logger::info ( const std::string & file, const int line, const std::string functionName, const std::string & message ) [virtual]`

Log a INFO **Level** (p. 2290) Log.

If the logger is currently enabled for the INFO message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2290).



6.488.3.18 `virtual bool decaf::util::logging::Logger::isLoggable ( const Level & level ) const`  
[virtual]

Check if a message of the given level would actually be logged by this logger.

This check is based on the Loggers effective level, which may be inherited from its parent.

#### Parameters

<i>level</i>	- a message logging level
--------------	---------------------------

#### Returns

true if the given message level is currently being logged.

6.488.3.19 `virtual void decaf::util::logging::Logger::log ( const Level & levels, const std::string & file, const int line, const std::string & message, ... )` [virtual]

Log a message, with the list of params that is formatted into the message string.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects

#### Parameters

<i>level</i>	the <b>Level</b> (p. 2290) to log at
<i>file</i>	the message to log
<i>line</i>	the line in the file
<i>...</i>	variable length argument to format the message string.

6.488.3.20 `virtual void decaf::util::logging::Logger::log ( LogRecord & record )`  
[virtual]

Log a **LogRecord** (p. 2370).

All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

#### Parameters

<i>record</i>	- the <b>LogRecord</b> (p. 2370) to be published
---------------	--

6.488.3.21 `virtual void decaf::util::logging::Logger::log ( const Level & level, const std::string & message )` [virtual]

Log a message, with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects

**Parameters**

<i>level</i>	the <b>Level</b> (p. 2290) to log at
<i>message</i>	the message to log

6.488.3.22 `virtual void decaf::util::logging::Logger::log ( const Level & level, const std::string & file, const int line, const std::string & message, lang::Exception & ex )`  
`[virtual]`

Log a message, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2370) which is forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 2370) thrown property, rather than the **LogRecord** (p. 2370) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 2370) message property.

**Parameters**

<i>level</i>	the <b>Level</b> (p. 2290) to log at.
<i>file</i>	File that the message was logged in.
<i>line</i>	the line number where the message was logged at.
<i>message</i>	the message to log.
<i>ex</i>	the Exception to log

6.488.3.23 `void decaf::util::logging::Logger::removeHandler ( Handler * handler )`

Removes the specified **Handler** (p. 1941) from this logger, ownership of the **Handler** (p. 1941) pointer is returned to the caller.

Returns silently if the given **Handler** (p. 1941) is not found.

**Parameters**

<i>handler</i>	The <b>Handler</b> (p. 1941) to remove
----------------	--

6.488.3.24 `void decaf::util::logging::Logger::setFilter ( Filter * filter )`

**Set** (p. 3379) a filter to control output on this **Logger** (p. 2345).

After passing the initial "level" check, the **Logger** (p. 2345) will call this **Filter** (p. 1853) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

**Parameters**

<i>filter</i>	The <b>Filter</b> (p. 1853) to use, (can be NULL).
---------------	--

6.488.3.25 void decaf::util::logging::Logger::setLevel ( const Level & *level* ) [inline]

**Set** (p. 3379) the log level specifying which message levels will be logged by this logger.

Message levels lower than this value will be discarded. The level value **Level::OFF** (p. 2295) can be used to turn off logging.

If the new level is the INHERIT **Level** (p. 2290), it means that this node should inherit its level from its nearest ancestor with a specific (non-INHERIT) level value.

#### Parameters

<i>level</i>	The new <b>Level</b> (p. 2290) value to use when logging.
--------------	---

6.488.3.26 void decaf::util::logging::Logger::setParent ( Logger \* *parent* ) [inline]

**Set** (p. 3379) the parent for this **Logger** (p. 2345).

This method is used by the **LogManager** (p. 2363) to update a **Logger** (p. 2345) when the namespace changes.

It should not be called from application code.

6.488.3.27 void decaf::util::logging::Logger::setUseParentHandlers ( bool *value* )  
[inline]

Specify whether or not this logger should send its output to it's parent **Logger** (p. 2345).

This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

#### Parameters

<i>value</i>	True is output is to be written to the parent
--------------	---

6.488.3.28 virtual void decaf::util::logging::Logger::severe ( const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message* ) [virtual]

Log a SEVERE **Level** (p. 2290) Log.

If the logger is currently enabled for the SEVERE message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2290).

6.488.3.29 `virtual void decaf::util::logging::Logger::throwing ( const std::string & file, const int line, const std::string functionName, const decaf::lang::Throwable & thrown )`  
`[virtual]`

Log throwing an exception.

This is a convenience method to log that a method is terminating by throwing an exception. The logging is done using the FINER level.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 2370) which is forwarded to all registered output handlers. The LogRecord's message is set to "THROW".

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>thrown</i>	The Throwable that will be thrown, will be cloned.

6.488.3.30 `virtual void decaf::util::logging::Logger::warning ( const std::string & file, const int line, const std::string functionName, const std::string & message )`  
`[virtual]`

Log a WARN **Level** (p. 2290) Log.

If the logger is currently enabled for the WARN message level then the given message is forwarded to all the registered output **Handler** (p. 1941) objects.

#### Parameters

<i>file</i>	The file name where the log was generated.
<i>line</i>	The line number where the log was generated.
<i>function-Name</i>	The name of the function that logged this.
<i>message</i>	The message to log at this <b>Level</b> (p. 2290).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Logger.h`

## 6.489 decaf::util::logging::LoggerHierarchy Class Reference

```
#include <src/main/decaf/util/logging/LoggerHierarchy.h>
```

### Public Member Functions

- `LoggerHierarchy ()`

- virtual `~LoggerHierarchy()`

### 6.489.1 Constructor & Destructor Documentation

6.489.1.1 `decaf::util::logging::LoggerHierarchy::LoggerHierarchy()`

6.489.1.2 `virtual decaf::util::logging::LoggerHierarchy::~LoggerHierarchy()` `[virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LoggerHierarchy.h`

## 6.490 activemq::io::LoggingInputStream Class Reference

```
#include <src/main/activemq/io/LoggingInputStream.h>
```

Inheritance diagram for `activemq::io::LoggingInputStream`:

### Public Member Functions

- **LoggingInputStream** (`decaf::io::InputStream *inputStream`, `bool own=false`)

*Creates a DataInputStream that uses the specified underlying InputStream.*

- virtual `~LoggingInputStream()`

### Protected Member Functions

- virtual `int doReadByte()` `throw ( decaf::io::IOException )`
- virtual `int doReadArrayBounded` (`unsigned char *buffer`, `int size`, `int offset`, `int length`) `throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

### 6.490.1 Constructor & Destructor Documentation

6.490.1.1 `activemq::io::LoggingInputStream::LoggingInputStream ( decaf::io::InputStream * inputStream, bool own = false )`

Creates a DataInputStream that uses the specified underlying InputStream.

#### Parameters

<i>inputStream</i>	the InputStream instance to wrap.
<i>own</i>	indicates if this class owns the wrapped string defaults to false.

6.490.1.2 `virtual activemq::io::LoggingInputStream::~~LoggingInputStream ( )`  
`[virtual]`

## 6.490.2 Member Function Documentation

6.490.2.1 `virtual int activemq::io::LoggingInputStream::doReadArrayBounded ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )` `[protected, virtual]`

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

6.490.2.2 `virtual int activemq::io::LoggingInputStream::doReadByte ( ) throw ( decaf::io::IOException )` `[protected, virtual]`

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingInputStream.h`

## 6.491 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

```
#include <src/main/activemq/io/LoggingOutputStream.h>
```

Inheritance diagram for `activemq::io::LoggingOutputStream`:

### Public Member Functions

- **LoggingOutputStream** (OutputStream \*next, bool own=false)  
*Constructor.*
- `virtual ~LoggingOutputStream ()`

### Protected Member Functions

- `virtual void doWriteByte` (unsigned char c) throw ( decaf::io::IOException )
- `virtual void doWriteArrayBounded` (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.491.1 Detailed Description

OutputStream filter that just logs the data being written.

### 6.491.2 Constructor & Destructor Documentation

6.491.2.1 `activemq::io::LoggingOutputStream::LoggingOutputStream ( OutputStream * next, bool own = false )`

Constructor.

#### Parameters

<i>next</i>	The OutputStream to wrap an write logs to.
<i>own</i>	If true, this object will control the lifetime of the output stream that it encapsulates.

6.491.2.2 `virtual activemq::io::LoggingOutputStream::~~LoggingOutputStream ( )`  
[virtual]

### 6.491.3 Member Function Documentation

6.491.3.1 `virtual void activemq::io::LoggingOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
[protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1863).

6.491.3.2 `virtual void activemq::io::LoggingOutputStream::doWriteByte ( unsigned char c ) throw ( decaf::io::IOException )` [protected, virtual]

Reimplemented from `decaf::io::FilterOutputStream` (p. 1863).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingOutputStream.h`

## 6.492 activemq::transport::logging::LoggingTransport Class Reference

A transport filter that logs commands as they are sent/received.

```
#include <src/main/activemq/transport/logging/LoggingTransport.h>
```

Inheritance diagram for activemq::transport::logging::LoggingTransport:

## Public Member Functions

- **LoggingTransport** (const **Pointer**< **Transport** > &next)

*Constructor.*

- virtual ~**LoggingTransport** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

*Event handler for the receipt of a command.*

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Sends a one-way command.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported by this class - throws an exception.*

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported by this class - throws an exception.*

## 6.492.1 Detailed Description

A transport filter that logs commands as they are sent/received.

## 6.492.2 Constructor & Destructor Documentation

### 6.492.2.1 activemq::transport::logging::LoggingTransport::LoggingTransport ( const **Pointer**< **Transport** > & next )

Constructor.

#### Parameters

<i>next</i>	- the next <b>Transport</b> (p. 3819) in the chain
-------------	--

### 6.492.2.2 virtual activemq::transport::logging::LoggingTransport::~~LoggingTransport ( ) [inline, virtual]

## 6.492.3 Member Function Documentation



6.492.3.1 virtual void activemq::transport::logging::LoggingTransport::onCommand ( const Pointer< Command > & command ) [virtual]

Event handler for the receipt of a command.

#### Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

6.492.3.2 virtual void activemq::transport::logging::LoggingTransport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

6.492.3.3 virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	the command that is sent as a request
----------------	---------------------------------------

#### Exceptions

<i>UnsupportedOperationException</i>	
--------------------------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 3833).

```

6.492.3.4 virtual Pointer<Response> ac-
tivemq::transport::logging::LoggingTransport::request (
const Pointer< Command > & command, unsigned
int timeout ) throw ( decaf::io::IOException, de-
caf::lang::exceptions::UnsupportedOperationException )
[virtual]

```

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	the command that is sent as a request
<i>timeout</i>	the time to wait for a response.

#### Exceptions

<i>UnsupportedOpera- tionException.</i>	
---	--

Reimplemented from **activemq::transport::TransportFilter** (p. 3833).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/logging/**LoggingTransport.h**

## 6.493 decaf::util::logging::LogManager Class Reference

There is a single global **LogManager** (p. 2363) object that is used to maintain a set of shared state about Loggers and log services.

```
#include <src/main/decaf/util/logging/LogManager.h>
```

#### Public Member Functions

- virtual **~LogManager** ()
- bool **addLogger** (**Logger** \*logger) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Add a named logger.*
- **Logger** \* **getLogger** (const std::string &name)  
*Retrieves or creates a new **Logger** (p. 2345) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*
- int **getLoggerNames** (const std::vector< std::string > &names)  
*Gets a list of known **Logger** (p. 2345) Names from this Manager, new loggers added while this method is in progress are not garunteed to be in the list.*
- void **setProperties** (const util::Properties &properties)  
*Sets the **Properties** (p. 3072) this **LogManager** (p. 2363) should use to configure its loggers.*

- const **util::Properties** & **getProperties** () const  
*Gets a reference to the Logging **Properties** (p. 3072) used by this logger.*
- std::string **getProperty** (const std::string &name)  
*Gets the value of a named property of this **LogManager** (p. 2363).*
- void **addPropertyChangeListener** (PropertyChangeListener \*listener)  
*Adds a change listener for **LogManager** (p. 2363) **Properties** (p. 3072), adding the same instance of a change event listener does nothing.*
- void **removePropertyChangeListener** (PropertyChangeListener \*listener)  
*Removes a properties change listener from the **LogManager** (p. 2363), if the listener is not found of the param is NULL this method returns silently.*
- void **readConfiguration** () throw ( decaf::io::IOException )  
*Reinitialize the logging properties and reread the logging configuration.*
- void **readConfiguration** (decaf::io::InputStream \*stream) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 3072) format.*
- void **reset** ()  
*Reset the logging configuration.*

### Static Public Member Functions

- static **LogManager** & **getLogManager** ()  
*Get the global **LogManager** (p. 2363) instance.*

### Protected Member Functions

- **LogManager** ()  
*Constructor, hidden to protect against direct instantiation.*
- **LogManager** (const **LogManager** &manager)  
*Copy Constructor.*
- void **operator=** (const **LogManager** &manager)  
*Assignment operator.*

### Friends

- class **decaf::lang::Runtime**

### 6.493.1 Detailed Description

There is a single global **LogManager** (p. 2363) object that is used to maintain a set of shared state about Loggers and log services.

This **LogManager** (p. 2363) object:

\* Manages a hierarchical namespace of **Logger** (p. 2345) objects. All named Loggers are stored in this namespace. \* Manages a set of logging control properties. These are simple key-value pairs that can be used by Handlers and other logging objects to configure themselves.

The global **LogManager** (p. 2363) object can be retrieved using **LogManager::getLogManager()** (p. 2368). The **LogManager** (p. 2363) object is created during class initialization and cannot subsequently be changed.

\*\*\*TODO\*\*\* By default, the **LogManager** (p. 2363) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default logging configuration for all uses of that JRE.

In addition, the **LogManager** (p. 2363) uses two optional system properties that allow more control over reading the initial configuration:

\* "decaf.logger.config.class" \* "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to JNI\_-CreateJavaVM.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use readConfiguration(InputStream) to define properties in the **LogManager** (p. 2363).

If "decaf.util.logging.config.class" property is not set, then the "decaf.util.logging.config.file" system property can be used to specify a properties file (in **decaf.util.Properties** (p. 3072) format). The initial logging configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p. 2363) will read its initial configuration from a properties file "lib/logging.properties" in the working directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. \*\*\*TODO\*\*\*

The global logging properties may include:

\* A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p. 2345) (the **Logger** (p. 2345) named ""). Each class name must be for a **Handler** (p. 1941) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. \* A property "<logger>.handlers". This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 1941) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. \* A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the logging message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 1941) needs to be configured for this logger otherwise no logging messages are delivered. \* A property "config". This property is

intended to allow arbitrary configuration code to be run. The property defines a white-space separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the logging configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and "a.b2" are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 2363) object are multi-thread safe.

### Since

1.0

## 6.493.2 Constructor & Destructor Documentation

6.493.2.1 `virtual decaf::util::logging::LogManager::~LogManager ( ) [virtual]`

6.493.2.2 `decaf::util::logging::LogManager::LogManager ( ) [protected]`

Constructor, hidden to protect against direct instantiation.

6.493.2.3 `decaf::util::logging::LogManager::LogManager ( const LogManager & manager ) [protected]`

Copy Constructor.

### Parameters

<i>manager</i>	the Manager to copy
----------------	---------------------

## 6.493.3 Member Function Documentation

6.493.3.1 `bool decaf::util::logging::LogManager::addLogger ( Logger * logger ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )`

Add a named logger.

This does nothing and returns false if a logger with the same name is already registered.

The **Logger** (p. 2345) factory methods call this method to register each newly created **Logger** (p. 2345).

**Parameters**

<i>logger</i>	The new <b>Logger</b> (p. 2345) instance to add to this <b>LogManager</b> (p. 2363).
---------------	--

**Exceptions**

<i>NullPointerException</i>	if logger is NULL.
<i>IllegalArgumentException</i>	if the logger has no name.

6.493.3.2 void decaf::util::logging::LogManager::addChangeListener (   
 PropertyChangeListener \* *listener* )

Adds a change listener for **LogManager** (p. 2363) **Properties** (p. 3072), adding the same instance of a change event listener does nothing.

**Parameters**

<i>listener</i>	The PropertyChangeListener to add (can be NULL).
-----------------	--

6.493.3.3 **Logger\*** decaf::util::logging::LogManager::getLogger ( const std::string & *name* )

Retrieves or creates a new **Logger** (p. 2345) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

**Parameters**

<i>name</i>	The name of the <b>Logger</b> (p. 2345).
-------------	--

6.493.3.4 int decaf::util::logging::LogManager::getLoggerNames ( const std::vector<   
 std::string > & *names* )

Gets a list of known **Logger** (p. 2345) Names from this Manager, new loggers added while this method is in progress are not guaranteed to be in the list.

**Parameters**

<i>names</i>	STL Vector to hold string logger names.
--------------	---

**Returns**

names count of how many loggers were inserted.

6.493.3.5 `static LogManager& decaf::util::logging::LogManager::getLogManager ( )`  
[static]

Get the global **LogManager** (p. 2363) instance.

#### Returns

A reference to the global **LogManager** (p. 2363) instance.

6.493.3.6 `const util::Properties& decaf::util::logging::LogManager::getProperties ( ) const`  
[inline]

Gets a reference to the Logging **Properties** (p. 3072) used by this logger.

#### Returns

The **Logger** (p. 2345) **Properties** (p. 3072) Pointer

6.493.3.7 `std::string decaf::util::logging::LogManager::getProperty ( const std::string & name )`

Gets the value of a named property of this **LogManager** (p. 2363).

#### Parameters

<i>name</i>	The name of the Property to retrieve.
-------------	---------------------------------------

#### Returns

the value of the property

6.493.3.8 `void decaf::util::logging::LogManager::operator= ( const LogManager & manager )`  
[protected]

Assignment operator.

#### Parameters

<i>manager</i>	the manager to assign from
----------------	----------------------------

6.493.3.9 `void decaf::util::logging::LogManager::readConfiguration ( ) throw ( decaf::io::IOException )`

Reinitialize the logging properties and reread the logging configuration.

The same rules are used for locating the configuration properties as are used at startup. So normally the logging properties will be re-read from the same file that was used at

startup.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 2355), if the target **Logger** (p. 2345) exists.

A **PropertyChangeEvent** will be fired after the properties are read.

### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

6.493.3.10 **void decaf::util::logging::LogManager::readConfiguration ( decaf::io::InputStream \* *stream* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )**

Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in **decaf.util.Properties** (p. 3072) format.

A **PropertyChangeEvent** will be fired after the properties are read.

Any log level definitions in the new configuration file will be applied using **Logger.setLevel()** (p. 2355), if the target **Logger** (p. 2345) exists.

### Parameters

<i>stream</i>	The <b>InputStream</b> to read the <b>Properties</b> (p. 3072) from.
---------------	--

### Exceptions

<i>NullPointerException</i>	if stream is NULL.
<i>IOException</i>	if an I/O error occurs.

6.493.3.11 **void decaf::util::logging::LogManager::removePropertyChangeListener ( PropertyChangeListener \* *listener* )**

Removes a properties change listener from the **LogManager** (p. 2363), if the listener is not found of the param is NULL this method returns silently.

### Parameters

<i>listener</i>	The <b>PropertyChangeListener</b> to remove from the listeners set.
-----------------	---

6.493.3.12 **void decaf::util::logging::LogManager::reset ( )**

Reset the logging configuration.

For all named loggers, the reset operation removes and closes all Handlers and (except for the root logger) sets the level to INHERIT. The root logger's level is set to **Level::INFO** (p. 2295).



6.493.3.13 void decaf::util::logging::LogManager::setProperties ( const util::Properties & properties )

Sets the **Properties** (p. 3072) this **LogManager** (p. 2363) should use to configure its loggers.

Once set a properties change event is fired.

#### Parameters

<i>properties</i>	Pointer to read the configuration from
-------------------	--

### 6.493.4 Friends And Related Function Documentation

6.493.4.1 friend class decaf::lang::Runtime [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogManager.h**

## 6.494 decaf::util::logging::LogRecord Class Reference

**LogRecord** (p. 2370) objects are used to pass logging requests between the logging framework and individual log Handlers.

```
#include <src/main/decaf/util/logging/LogRecord.h>
```

### Public Member Functions

- **LogRecord** ()
- virtual **~LogRecord** ()
- **Level** **getLevel** () const  
*Get **Level** (p. 2290) of this log record.*
- void **setLevel** (**Level** value)  
***Set** (p. 3379) the **Level** (p. 2290) of this Log Record.*
- const std::string & **getLoggerName** () const  
*Gets the Source Logger's Name.*
- void **setLoggerName** (const std::string &loggerName)  
*Sets the Source Logger's Name.*
- const std::string & **getSourceFile** () const  
*Gets the Source Log File name.*
- void **setSourceFile** (const std::string &sourceFile)  
*Sets the Source Log File Name.*
- unsigned int **getSourceLine** () const  
*Gets the Source Log line number.*

- void **setSourceLine** (unsigned int sourceLine)  
*Sets the Source Log line number.*
- const std::string & **getMessage** () const  
*Gets the Message to be Logged.*
- void **setMessage** (const std::string &message)  
*Sets the Message to be Logged.*
- const std::string & **getSourceFunction** () const  
*Gets the name of the function where this log was logged.*
- void **setSourceFunction** (const std::string &functionName)  
*Sets the name of the function where this log was logged.*
- long long **getTimestamp** () const  
*Gets the time in mills that this message was logged.*
- void **setTimestamp** (long long timeStamp)  
*Sets the time in mills that this message was logged.*
- long long **getTreadId** () const  
*Gets the Thread Id where this Log was created.*
- void **setTreadId** (long long threadId)  
*Sets the Thread Id where this Log was created.*
- **decaf::lang::Throwable** \* **getThrown** () const  
*Gets any Throwable associated with this **LogRecord** (p. 2370).*
- void **setThrown** (**decaf::lang::Throwable** \*thrown)  
*Sets the Throwable associated with this **LogRecord** (p. 2370), the pointer becomes the property of this instance of the **LogRecord** (p. 2370) and will be deleted when the record is destroyed.*

#### 6.494.1 Detailed Description

**LogRecord** (p. 2370) objects are used to pass logging requests between the logging framework and individual log Handlers.

When a **LogRecord** (p. 2370) is passed into the logging framework it logically belongs to the framework and should no longer be used or updated by the client application.

Since

1.0

#### 6.494.2 Constructor & Destructor Documentation

6.494.2.1 **decaf::util::logging::LogRecord::LogRecord** ( )

6.494.2.2 **virtual decaf::util::logging::LogRecord::~~LogRecord** ( ) [virtual]

#### 6.494.3 Member Function Documentation

6.494.3.1 **Level** decaf::util::logging::LogRecord::getLevel ( ) const [inline]

Get **Level** (p. 2290) of this log record.

#### Returns

**Level** (p. 2290) enumeration value.

6.494.3.2 **const std::string&** decaf::util::logging::LogRecord::getLoggerName ( ) const [inline]

Gets the Source Logger's Name.

#### Returns

the source loggers name

6.494.3.3 **const std::string&** decaf::util::logging::LogRecord::getMessage ( ) const [inline]

Gets the Message to be Logged.

#### Returns

the source logger's message

6.494.3.4 **const std::string&** decaf::util::logging::LogRecord::getSourceFile ( ) const [inline]

Gets the Source Log File name.

#### Returns

the source loggers name

6.494.3.5 **const std::string&** decaf::util::logging::LogRecord::getSourceFunction ( ) const [inline]

Gets the name of the function where this log was logged.

#### Returns

the source logger's message

6.494.3.6 `unsigned int decaf::util::logging::LogRecord::getSourceLine ( ) const [inline]`

Gets the Source Log line number.

#### Returns

the source loggers line number

6.494.3.7 `decaf::lang::Throwable* decaf::util::logging::LogRecord::getThrown ( ) const [inline]`

Gets any Throwable associated with this **LogRecord** (p. 2370).

#### Returns

point to a Throwable instance or Null.

6.494.3.8 `long long decaf::util::logging::LogRecord::getTimestamp ( ) const [inline]`

Gets the time in mills that this message was logged.

#### Returns

UTC time in milliseconds

6.494.3.9 `long long decaf::util::logging::LogRecord::getTreadId ( ) const [inline]`

Gets the Thread Id where this Log was created.

#### Returns

the source loggers line number

6.494.3.10 `void decaf::util::logging::LogRecord::setLevel ( Level value ) [inline]`

**Set** (p. 3379) the **Level** (p. 2290) of this Log Record.

#### Parameters

<i>value</i>	<b>Level</b> (p. 2290) Enumeration Value
--------------	--

6.494.3.11 void decaf::util::logging::LogRecord::setLoggerName ( const std::string & *loggerName* ) [inline]

Sets the Source Logger's Name.

#### Parameters

<i>loggerName</i>	the source loggers name
-------------------	-------------------------

6.494.3.12 void decaf::util::logging::LogRecord::setMessage ( const std::string & *message* ) [inline]

Sets the Message to be Logged.

#### Parameters

<i>message</i>	the source loggers message
----------------	----------------------------

6.494.3.13 void decaf::util::logging::LogRecord::setSourceFile ( const std::string & *sourceFile* ) [inline]

Sets the Source Log File Name.

#### Parameters

<i>sourceFile</i>	the source loggers name
-------------------	-------------------------

6.494.3.14 void decaf::util::logging::LogRecord::setSourceFunction ( const std::string & *functionName* ) [inline]

Sets the name of the function where this log was logged.

#### Parameters

<i>function-Name</i>	the source of the log
----------------------	-----------------------

6.494.3.15 void decaf::util::logging::LogRecord::setSourceLine ( unsigned int *sourceLine* ) [inline]

Sets the Source Log line number.

#### Parameters

<i>sourceLine</i>	the source logger's line number
-------------------	---------------------------------

6.494.3.16 `void decaf::util::logging::LogRecord::setThrown ( decaf::lang::Throwable *  
thrown ) [inline]`

Sets the Throwable associated with this **LogRecord** (p. 2370), the pointer becomes the property of this instance of the **LogRecord** (p. 2370) and will be deleted when the record is destroyed.

#### Parameters

<i>thrown</i>	A pointer to a Throwable that will be associated with this record.
---------------	--

6.494.3.17 `void decaf::util::logging::LogRecord::setTimestamp ( long long timeStamp )  
[inline]`

Sets the time in mills that this message was logged.

#### Parameters

<i>timeStamp</i>	UTC Time in Milliseconds.
------------------	---------------------------

6.494.3.18 `void decaf::util::logging::LogRecord::setTreadId ( long long threadId )  
[inline]`

Sets the Thread Id where this Log was created.

#### Parameters

<i>threadId</i>	the source logger's line number
-----------------	---------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogRecord.h**

## 6.495 decaf::util::logging::LogWriter Class Reference

```
#include <src/main/decaf/util/logging/LogWriter.h>
```

### Public Member Functions

- **LogWriter** ()
- virtual **~LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)

*Writes a message to the output destination.*

- virtual void **log** (const std::string &message)

*Writes a message to the output destination.*

### Static Public Member Functions

- static **LogWriter** & **getInstance** ()

*Get the singleton instance.*

- static void **returnInstance** ()

*Returns a Checked out instance of this Writer.*

- static void **destroy** ()

*Forcefully Delete the Instance of this **LogWriter** (p. 2375) even if there are outstanding references.*

### 6.495.1 Constructor & Destructor Documentation

6.495.1.1 decaf::util::logging::LogWriter::LogWriter ( )

6.495.1.2 virtual decaf::util::logging::LogWriter::~~LogWriter ( ) [virtual]

### 6.495.2 Member Function Documentation

6.495.2.1 static void decaf::util::logging::LogWriter::destroy ( ) [static]

Forcefully Delete the Instance of this **LogWriter** (p. 2375) even if there are outstanding references.

6.495.2.2 static **LogWriter**& decaf::util::logging::LogWriter::getInstance ( ) [static]

Get the singleton instance.

6.495.2.3 virtual void decaf::util::logging::LogWriter::log ( const std::string & message )  
[virtual]

Writes a message to the output destination.

#### Parameters

<i>message</i>	
----------------	--

6.495.2.4 virtual void decaf::util::logging::LogWriter::log ( const std::string & file, const int line, const std::string & prefix, const std::string & message ) [virtual]

Writes a message to the output destination.

**Parameters**

<i>file</i>	
<i>line</i>	
<i>prefix</i>	
<i>message</i>	

6.495.2.5 `static void decaf::util::logging::LogWriter::returnInstance ( ) [static]`

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LogWriter.h`

## 6.496 decaf::lang::Long Class Reference

```
#include <src/main/decaf/lang/Long.h>
```

Inheritance diagram for decaf::lang::Long:

**Public Member Functions**

- **Long** (long long value)
- **Long** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual **~Long** ()
- virtual int **compareTo** (const **Long** &l) const  
*Compares this **Long** (p. 2377) instance with another.*
- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Long** &l) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual int **compareTo** (const long long &l) const  
*Compares this **Long** (p. 2377) instance with another.*
- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const long long &l) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **toString** () const



- virtual double **doubleValue** () const  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const  
*Answers the float value which the receiver represents.*
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual short **shortValue** () const  
*Answers the short value which the receiver represents.*
- virtual int **intValue** () const  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const  
*Answers the long value which the receiver represents.*

### Static Public Member Functions

- static int **bitCount** (long long value)  
*Returns the number of one-bits in the two's complement binary representation of the specified int value.*
- static **Long decode** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Decodes a **String** (p. 3610) into a **Long** (p. 2377).*
- static long long **highestOneBit** (long long value)  
*Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.*
- static long long **lowestOneBit** (long long value)  
*Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.*
- static int **numberOfLeadingZeros** (long long value)  
*Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.*
- static int **numberOfTrailingZeros** (long long value)  
*Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.*
- static long long **parseLong** (const std::string &value) throw ( exceptions::NumberFormatException )  
*Parses the string argument as a signed decimal long.*
- static long long **parseLong** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )  
*Returns a **Long** (p. 2377) object holding the value extracted from the specified string when parsed with the radix given by the second argument.*
- static long long **reverseBytes** (long long value)  
*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.*
- static long long **reverse** (long long value)

*Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.*

- static long long **rotateLeft** (long long value, int distance)

*Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.*

- static long long **rotateRight** (long long value, int distance)

*Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.*

- static int **signum** (long long value)

*Returns the signum function of the specified value.*

- static std::string **toString** (long long value)

*Converts the long to a **String** (p. 3610) representation.*

- static std::string **toString** (long long value, int radix)

- static std::string **toHexString** (long long value)

*Returns a string representation of the integer argument as an unsigned integer in base 16.*

- static std::string **toOctalString** (long long value)

*Returns a string representation of the long long argument as an unsigned long long in base 8.*

- static std::string **toBinaryString** (long long value)

*Returns a string representation of the long long argument as an unsigned long long in base 2.*

- static **Long valueOf** (long long value)

*Returns a **Long** (p. 2377) instance representing the specified int value.*

- static **Long valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )

*Returns a **Long** (p. 2377) object holding the value given by the specified std::string.*

- static **Long valueOf** (const std::string &value, int radix) throw ( exceptions::NumberFormatException )

*Returns a **Long** (p. 2377) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

## Static Public Attributes

- static const int **SIZE** = 64

*The size in bits of the primitive long long type.*

- static const long long **MAX\_VALUE** = (long long)0x7FFFFFFFFFFFFFFFFLL

*The maximum value that the primitive type can hold.*

- static const long long **MIN\_VALUE** = (long long)0x8000000000000000LL

*The minimum value that the primitive type can hold.*

## 6.496.1 Constructor & Destructor Documentation

6.496.1.1 `decaf::lang::Long::Long ( long long value )`

### Parameters

<i>value</i>	- the primitive long long to wrap
--------------	-----------------------------------

6.496.1.2 `decaf::lang::Long::Long ( const std::string & value ) throw ( exceptions::NumberFormatException )`

### Parameters

<i>value</i>	- the long long formatted string to wrap
--------------	--

### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

6.496.1.3 `virtual decaf::lang::Long::~Long ( ) [inline, virtual]`

## 6.496.2 Member Function Documentation

6.496.2.1 `static int decaf::lang::Long::bitCount ( long long value ) [static]`

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

### Parameters

<i>value</i>	- the long long to count
--------------	--------------------------

### Returns

the number of one-bits in the two's complement binary representation of the specified long long value.

6.496.2.2 `virtual unsigned char decaf::lang::Long::byteValue ( ) const [inline, virtual]`

Answers the byte value which the receiver represents.

### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2787).

6.496.2.3 `virtual int decaf::lang::Long::compareTo ( const long long & / ) const` [virtual]

Compares this **Long** (p. 2377) instance with another.

#### Parameters

<i>/</i> - the <b>Integer</b> (p. 2038) instance to be compared
---

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **long long** > (p. 1187).

6.496.2.4 `virtual int decaf::lang::Long::compareTo ( const Long & / ) const` [virtual]

Compares this **Long** (p. 2377) instance with another.

#### Parameters

<i>/</i> - the <b>Long</b> (p. 2377) instance to be compared
--

#### Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Long** > (p. 1187).

6.496.2.5 `static Long decaf::lang::Long::decode ( const std::string & value ) throw ( exceptions::NumberFormatException )` [static]

Decodes a **String** (p. 3610) into a **Long** (p. 2377).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified **String** (p. 3610) is the minus sign. No whitespace characters are permitted in the string.

#### Parameters

<i>value</i> - The string to decode
-------------------------------------

**Returns**

a **Long** (p. 2377) object containing the decoded value

**Exceptions**

<i>NumberFormatException</i>	if the string is not formatted correctly.
------------------------------	---

6.496.2.6 `virtual double decaf::lang::Long::doubleValue ( ) const [inline, virtual]`

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.496.2.7 `bool decaf::lang::Long::equals ( const Long & / ) const [inline, virtual]`

**Parameters**

/	- the <b>Long</b> (p. 2377) object to compare against.
---	--

**Returns**

true if the two **Integer** (p. 2038) Objects have the same value.

Implements **decaf::lang::Comparable**< **Long** > (p. 1188).

6.496.2.8 `bool decaf::lang::Long::equals ( const long long & / ) const [inline, virtual]`

**Parameters**

/	- the <b>Long</b> (p. 2377) object to compare against.
---	--

**Returns**

true if the two **Integer** (p. 2038) Objects have the same value.

Implements **decaf::lang::Comparable**< **long long** > (p. 1188).

6.496.2.9 `virtual float decaf::lang::Long::floatValue ( ) const [inline, virtual]`

Answers the float value which the receiver represents.

**Returns**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.496.2.10 `static long long decaf::lang::Long::highestOneBit ( long long value ) [static]`

Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Parameters**

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

**Returns**

an long long value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.496.2.11 `virtual int decaf::lang::Long::intValue ( ) const [inline, virtual]`

Answers the int value which the receiver represents.

**Returns**

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.496.2.12 `virtual long long decaf::lang::Long::longValue ( ) const [inline, virtual]`

Answers the long value which the receiver represents.

**Returns**

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.496.2.13 `static long long decaf::lang::Long::lowestOneBit ( long long value ) [static]`

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Parameters**

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

**Returns**

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

**6.496.2.14** `static int decaf::lang::Long::numberOfLeadingZeros ( long long value )`  
[static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values *x*:

$\ast \text{floor}(\log_2(x)) = 63 - \text{numberOfLeadingZeros}(x)$   $\ast \text{ceil}(\log_2(x)) = 64 - \text{numberOfLeadingZeros}(x - 1)$

**Parameters**

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

**Returns**

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

**6.496.2.15** `static int decaf::lang::Long::numberOfTrailingZeros ( long long value )`  
[static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

**Parameters**

<i>value</i>	- the int to be inspected
--------------	---------------------------

**Returns**

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.496.2.16 `virtual bool decaf::lang::Long::operator< ( const long long & / ) const`  
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

/	- the value to be compared to this one.
---	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< long long >** (p. 1188).

6.496.2.17 `virtual bool decaf::lang::Long::operator< ( const Long & / ) const` [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

/	- the value to be compared to this one.
---	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Long >** (p. 1188).

6.496.2.18 `virtual bool decaf::lang::Long::operator==( const Long & / ) const` [inline, virtual]

Compares equality between this object and the one passed.

#### Parameters

/	- the value to be compared to this one.
---	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Long >** (p. 1189).



6.496.2.19 `virtual bool decaf::lang::Long::operator==( const long long & / ) const`  
[inline, virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>/</i>	- the value to be compared to this one.
----------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< long long > (p. 1189).

6.496.2.20 `static long long decaf::lang::Long::parseLong ( const std::string & value ) throw (`  
`exceptions::NumberFormatException )` [static]

Parses the string argument as a signed decimal long.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseLong(java.lang.String, int)` method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

#### Parameters

<i>value</i>	- <b>String</b> (p. 3610) to parse
--------------	------------------------------------

#### Returns

long long value

#### Exceptions

<i>NumberFormatException</i>	on invalid string value
------------------------------	-------------------------

6.496.2.21 `static long long decaf::lang::Long::parseLong ( const std::string & value, int radix )`  
`throw ( exceptions::NumberFormatException )` [static]

Returns a **Long** (p. 2377) object holding the value extracted from the specified string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 2377) object that represents the long long value specified by the string.

**Parameters**

<i>value</i>	- <b>String</b> (p. 3610) to parse
<i>radix</i>	- the base encoding of the string

**Returns**

long long value

**Exceptions**

<i>NumberFormatException</i>	on invalid string value
------------------------------	-------------------------

6.496.2.22 `static long long decaf::lang::Long::reverse ( long long value ) [static]`

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

**Parameters**

<i>value</i>	- the value whose bits are to be reversed
--------------	---

**Returns**

the reversed bits long long.

6.496.2.23 `static long long decaf::lang::Long::reverseBytes ( long long value ) [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

**Parameters**

<i>value</i>	- the long long whose bytes we are to reverse
--------------	---

**Returns**

the reversed long long.

6.496.2.24 `static long long decaf::lang::Long::rotateLeft ( long long value, int distance ) [static]`

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: `rotateLeft(val, -distance) == rotateRight(val, distance)`. Note also that rotation by any multiple of 32 is

a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F)`.

#### Parameters

<i>value</i>	- the long long to be inspected
<i>distance</i>	- the number of bits to rotate

#### Returns

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

**6.496.2.25** `static long long decaf::lang::Long::rotateRight ( long long value, int distance )`  
[static]

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: `rotateRight(val, -distance) == rotateLeft(val, distance)`. Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: `rotateRight(val, distance) == rotateRight(val, distance & 0x1F)`.

#### Parameters

<i>value</i>	- the long long to be inspected
<i>distance</i>	- the number of bits to rotate

#### Returns

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

**6.496.2.26** `virtual short decaf::lang::Long::shortValue ( ) const` [inline, virtual]

Answers the short value which the receiver represents.

#### Returns

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p.2788).

**6.496.2.27** `static int decaf::lang::Long::signum ( long long value )` [static]

Returns the signum function of the specified value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

#### Parameters

<i>value</i>	- the long long to be inspected
--------------	---------------------------------

#### Returns

the signum function of the specified long long value.

**6.496.2.28** `static std::string decaf::lang::Long::toBinaryString ( long long value )`  
[static]

Returns a string representation of the long long argument as an unsigned long long in base 2.

The unsigned long long value is the argument plus  $2^{32}$  if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

#### Parameters

<i>value</i>	- the long long to be translated to a binary string
--------------	---

#### Returns

the unsigned long long value as a binary string

**6.496.2.29** `static std::string decaf::lang::Long::toHexString ( long long value )` [static]

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

#### Parameters

<i>value</i>	- the long long to be translated to an Octal string
--------------	---

**Returns**

the unsigned long long value as a Octal string

6.496.2.30 `static std::string decaf::lang::Long::toOctalString ( long long value ) [static]`

Returns a string representation of the long long argument as an unsigned long long in base 8.

The unsigned long long value is the argument plus  $2^{32}$  if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

**Parameters**

<i>value</i>	- the long long to be translated to an Octal string
--------------	---

**Returns**

the unsigned long long value as a Octal string

6.496.2.31 `static std::string decaf::lang::Long::toString ( long long value ) [static]`

Converts the long to a **String** (p. 3610) representation.

**Parameters**

<i>value</i>	The long to convert to a std::string.
--------------	---------------------------------------

**Returns**

string representation

6.496.2.32 `std::string decaf::lang::Long::toString ( ) const`

**Returns**

this **Long** (p. 2377) Object as a **String** (p. 3610) Representation

6.496.2.33 `static std::string decaf::lang::Long::toString ( long long value, int radix ) [static]`

6.496.2.34 **static Long** decaf::lang::Long::valueOf ( long long *value* ) [inline, static]

Returns a **Long** (p. 2377) instance representing the specified int value.

#### Parameters

<i>value</i>	- the long long to wrap
--------------	-------------------------

#### Returns

the new **Integer** (p. 2038) object wrapping value.

6.496.2.35 **static Long** decaf::lang::Long::valueOf ( const std::string & *value*, int *radix* ) throw ( exceptions::NumberFormatException ) [static]

Returns a **Long** (p. 2377) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the parseLong( std::string, int ) method. The result is a **Long** (p. 2377) object that represents the long long value specified by the string.

#### Parameters

<i>value</i>	- std::string to parse as base ( radix )
<i>radix</i>	- base of the string to parse.

#### Returns

new **Long** (p. 2377) Object wrapping the primitive

#### Exceptions

<i>NumberFormatException</i>	if the string is not a valid long long.
------------------------------	---

6.496.2.36 **static Long** decaf::lang::Long::valueOf ( const std::string & *value* ) throw ( exceptions::NumberFormatException ) [static]

Returns a **Long** (p. 2377) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the parseLong( std::string ) method. The result is a **Integer** (p. 2038) object that represents the long long value specified by the string.

#### Parameters

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

Returns

new **Long** (p. 2377) Object wrapping the primitive

Exceptions

<i>NumberFormatException</i>	if the string is not a decimal long long.
------------------------------	---

6.496.3 Field Documentation

6.496.3.1 `const long long decaf::lang::Long::MAX_VALUE = (long long)0x7FFFFFFFFFFFFFFFLL` [static]

The maximum value that the primitive type can hold.

6.496.3.2 `const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL` [static]

The minimum value that the primitive type can hold.

6.496.3.3 `const int decaf::lang::Long::SIZE = 64` [static]

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Long.h**

6.497 decaf::internal::nio::LongArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/LongArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

Public Member Functions

- **LongArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **IntArrayBuffer** (p. 2015) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **LongArrayBuffer** (long long \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **LongArrayBuffer** (p. 2392) object that wraps the given array.

- **LongArrayBuffer** (const **decaf::lang::Pointer**< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException** )

Creates a byte buffer that wraps the passed **ByteArrayAdapter** and start at the given offset.

- **LongArrayBuffer** (const **LongArrayBuffer** &other)

Create a **LongArrayBuffer** (p. 2392) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayAdapter** and when changes are made to that data it is reflected in both.

- virtual ~**LongArrayBuffer** ()
- virtual long long \* **array** () throw ( **decaf::lang::exceptions::UnsupportedOperationException**, **decaf::nio::ReadOnlyBufferException** )

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 887).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b>UnsupportedOperationException</b>	if the underlying store has no array.

- virtual int **arrayOffset** () throw ( **decaf::lang::exceptions::UnsupportedOperationException**, **decaf::nio::ReadOnlyBufferException** )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the **hasArray** method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset long longo the backing array where index zero starts.

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b>UnsupportedOperationException</b>	if the underlying store has no array.

- virtual **LongBuffer** \* **asReadOnlyBuffer** () const

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as



the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only long long buffer which the caller then owns.

- virtual LongBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **LongBuffer** (p. 2403).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

- virtual LongBuffer \* **duplicate** ()

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new long long **Buffer** (p. 887) which the caller owns.

- virtual long long **get** () throw ( decaf::nio::BufferUnderflowException )

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the long long at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

- virtual long long **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

Absolute get method.

Reads the value at the given index.

#### Parameters

index	The index in the <b>Buffer</b> (p. 887) where the long long is to be read.
-------	--

**Returns**

*the long long that is located at the given index.*

**Exceptions**

IndexOutOfBoundsException	<i>if index is not smaller than the buffer's limit, or index is negative.</i>
---------------------------	---

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible long long array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only.*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual LongBuffer & **put** (long long value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given long longs long longo this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The long longs value to be written.</i>
-------	--

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	<i>if this buffer's current position is not smaller than its limit</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only</i>

- virtual LongBuffer & **put** (int index, long long value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given long longs long longo this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 887) to write the data</i>
value	<i>The long longs to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written.</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only</i>

- virtual LongBuffer \* **slice** () const

Creates a new **LongBuffer** (p. 2403) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **LongBuffer** (p. 2403) which the caller owns.

## Protected Member Functions

- virtual void **setReadOnly** (bool value)

Sets this **LongArrayBuffer** (p. 2392) as Read-Only.

## 6.497.1 Constructor & Destructor Documentation

6.497.1.1 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )

Creates a **IntArrayBuffer** (p. 2015) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

size	The size of the array, this is the limit we read and write to.
readOnly	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

IllegalArgumentException	if the capacity value is negative.
--------------------------	------------------------------------

6.497.1.2 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( long long \* array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

Creates a **LongArrayBuffer** (p. 2392) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

array	The actual array to wrap.
-------	---------------------------

<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.497.1.3 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **LongArrayBuffer** (p. 2392) will be that of the remaining capacity of the passed buffer.

### Parameters

<i>array</i>	The ByteArrayAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.497.1.4 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer ( const LongArrayBuffer & other )`

Create a **LongArrayBuffer** (p. 2392) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.

### Parameters

<i>other</i>	The <b>LongArrayBuffer</b> (p. 2392) this one is to mirror.
--------------	---

6.497.1.5 virtual decaf::internal::nio::LongArrayBuffer::~~LongArrayBuffer ( ) [virtual]

## 6.497.2 Member Function Documentation

6.497.2.1 virtual long long\* decaf::internal::nio::LongArrayBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

the array that backs this **Buffer** (p. 887).

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2406).

6.497.2.2 virtual int decaf::internal::nio::LongArrayBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset long longo the backing array where index zero starts.

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 2407).

6.497.2.3 **virtual LongBuffer\* decaf::internal::nio::LongArrayBuffer::asReadOnlyBuffer ( )**  
**const** [virtual]

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 2407).

6.497.2.4 **virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact ( ) throw (**  
**decaf::nio::ReadOnlyBufferException )** [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n + 1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **LongBuffer** (p. 2403).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::LongBuffer** (p. 2407).

6.497.2.5 `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::duplicate ( )`  
[virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

a new long long **Buffer** (p. 887) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2408).

6.497.2.6 `virtual long long decaf::internal::nio::LongArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException )` [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the long long at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::LongBuffer** (p. 2410).

6.497.2.7 `virtual long long decaf::internal::nio::LongArrayBuffer::get ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException )` [virtual]

Absolute get method.

Reads the value at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the long long is to be read.
--------------	--

#### Returns

the long long that is located at the given index.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or index is negative.
----------------------------------	--

Implements **decaf::nio::LongBuffer** (p. 2409).

6.497.2.8 `virtual bool decaf::internal::nio::LongArrayBuffer::hasArray ( ) const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::LongBuffer** (p. 2410).

6.497.2.9 `virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly ( ) const [inline, virtual]`

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 890).

6.497.2.10 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put ( int index, long long value ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException ) [virtual]`

Writes the given long longs long longo this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The long longs to write.

**Returns**

a reference to this buffer.



**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2411).

6.497.2.11 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put ( long long value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException ) [virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The long longs value to be written.
--------------	-------------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 2411).

6.497.2.12 virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly ( bool value ) [inline, protected, virtual]

Sets this **LongArrayBuffer** (p. 2392) as Read-Only.

**Parameters**

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.497.2.13 `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice ( ) const`  
`[virtual]`

Creates a new **LongBuffer** (p. 2403) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **LongBuffer** (p. 2403) which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 2414).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/LongArrayBuffer.h`

## 6.498 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:

```
#include <src/main/decaf/nio/LongBuffer.h>
```

Inheritance diagram for `decaf::nio::LongBuffer`:

### Public Member Functions

- `virtual ~LongBuffer ()`
- `virtual std::string toString () const`
- `virtual long long * array ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the long long array that backs this buffer (optional operation).*
- `virtual int arrayOffset ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )`  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- `virtual LongBuffer * asReadOnlyBuffer () const =0`  
*Creates a new, read-only long long buffer that shares this buffer's content.*
- `virtual LongBuffer & compact ()=0 throw ( ReadOnlyBufferException )`  
*Compacts this buffer.*

- virtual **LongBuffer** \* **duplicate** ()=0  
*Creates a new long long buffer that shares this buffer's content.*
- virtual long long **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual long long **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **LongBuffer** & **get** (std::vector< long long > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **LongBuffer** & **get** (long long \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible long long array.*
- **LongBuffer** & **put** (**LongBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException )  
*This method transfers the long longs remaining in the given source buffer long longo this buffer.*
- **LongBuffer** & **put** (const long long \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers long longs long longo this buffer from the given source array.*
- **LongBuffer** & **put** (std::vector< long long > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source long longs array long longo this buffer.*
- virtual **LongBuffer** & **put** (long long value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given long longs long longo this buffer at the current position, and then increments the position.*
- virtual **LongBuffer** & **put** (int index, long long value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given long longs long longo this buffer at the given index.*
- virtual **LongBuffer** \* **slice** () const =0  
*Creates a new **LongBuffer** (p. 2403) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **LongBuffer** &value) const
- virtual bool **equals** (const **LongBuffer** &value) const
- virtual bool **operator==** (const **LongBuffer** &value) const
- virtual bool **operator<** (const **LongBuffer** &value) const

### Static Public Member Functions

- static **LongBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new Double buffer.*
- static **LongBuffer** \* **wrap** (long long \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **LongBuffer** (p. 2403).*
- static **LongBuffer** \* **wrap** (std::vector< long long > &buffer)  
*Wraps the passed STL long long Vector in a **LongBuffer** (p. 2403).*

### Protected Member Functions

- **LongBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **LongBuffer** (p. 2403) object that has its backing array allocated long long-termally and is then owned and deleted when this object is deleted.*

#### 6.498.1 Detailed Description

This class defines four categories of operations upon long long buffers:

o Absolute and relative get and put methods that read and write single long longs; o Relative bulk get methods that transfer contiguous sequences of long longs from this buffer long long to an array; and o Relative bulk put methods that transfer contiguous sequences of long longs from a long long array or some other long long buffer long long to this buffer o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing long long array long long to a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

#### 6.498.2 Constructor & Destructor Documentation

- 6.498.2.1 decaf::nio::LongBuffer::LongBuffer ( int *capacity* ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [protected]

Creates a **LongBuffer** (p. 2403) object that has its backing array allocated long long-termally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p. 887) in long longs.
-----------------	---

**Exceptions**

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

6.498.2.2 virtual decaf::nio::LongBuffer::~LongBuffer ( ) [inline, virtual]

**6.498.3 Member Function Documentation**

6.498.3.1 static LongBuffer\* decaf::nio::LongBuffer::allocate ( int *capacity* ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

**Parameters**

<i>capacity</i>	The size of the Double buffer in long longs.
-----------------	--

**Returns**

the **LongBuffer** (p. 2403) that was allocated, caller owns.

6.498.3.2 virtual long long\* decaf::nio::LongBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 887).

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2398).

6.498.3.3 `virtual int decaf::nio::LongBuffer::arrayOffset ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException ) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset long long the backing array where index zero starts.

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<i>UnsupportedOperationException</i>	if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2398).

6.498.3.4 `virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer ( ) const [pure virtual]`

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only long long buffer which the caller then owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2399).

6.498.3.5 `virtual LongBuffer& decaf::nio::LongBuffer::compact ( ) throw ( ReadOnlyBufferException ) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

### Returns

a reference to this **LongBuffer** (p. 2403).

### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2399).

6.498.3.6 `virtual int decaf::nio::LongBuffer::compareTo ( const LongBuffer & value ) const`  
[virtual]

6.498.3.7 `virtual LongBuffer* decaf::nio::LongBuffer::duplicate ( )` [pure virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new long long **Buffer** (p. 887) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2400).

6.498.3.8 `virtual bool decaf::nio::LongBuffer::equals ( const LongBuffer & value ) const`  
[virtual]

6.498.3.9 `LongBuffer& decaf::nio::LongBuffer::get ( std::vector< long long > buffer ) throw ( BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer long longo the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this **Buffer** (p. 887).

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length long longs remaining in this buffer.
--	---

6.498.3.10 `virtual long long decaf::nio::LongBuffer::get ( int index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]`

Absolute get method.

Reads the value at the given index.

### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the long long is to be read.
--------------	--

### Returns

the long long that is located at the given index.

### Exceptions

<b><i>IndexOutOfBoundsException</i></b>	if index is not smaller than the buffer's limit, or index is negative.
---	--

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2400).

6.498.3.11 `LongBuffer& decaf::nio::LongBuffer::get ( long long * buffer, int size, int offset, int length ) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Relative bulk get method.

This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 892), then no bytes are transferred and a **BufferUnderflowException** (p. 916) is thrown.

Otherwise, this method copies length long longs from this buffer long longo the given



array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The pointer to an allocated long long buffer to fill.
<i>size</i>	The size of the passed in buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 887).

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there are fewer than length long longs remaining in this buffer
<i>NullPolong</i>	longerException if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.498.3.12 `virtual long long decaf::nio::LongBuffer::get ( ) throw ( BufferUnderflowException ) [pure virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the long long at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2400).

6.498.3.13 `virtual bool decaf::nio::LongBuffer::hasArray ( ) const [pure virtual]`

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2401).

6.498.3.14 `virtual bool decaf::nio::LongBuffer::operator< ( const LongBuffer & value ) const`  
[virtual]

6.498.3.15 `virtual bool decaf::nio::LongBuffer::operator== ( const LongBuffer & value ) const`  
[virtual]

6.498.3.16 `virtual LongBuffer& decaf::nio::LongBuffer::put ( long long value ) throw (`  
**BufferOverflowException, ReadOnlyBufferException** ) [pure  
virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

**Parameters**

<i>value</i>	The long longs value to be written.
--------------	-------------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2402).

6.498.3.17 `virtual LongBuffer& decaf::nio::LongBuffer::put ( int index, long long value`  
`) throw ( decaf::lang::exceptions::IndexOutOfBoundsException,`  
**ReadOnlyBufferException** ) [pure virtual]

Writes the given long longs long longo this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data
<i>value</i>	The long longs to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2401).

6.498.3.18 **LongBuffer& decaf::nio::LongBuffer::put** ( const long long \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **BufferOverflowException**, **ReadOnlyBufferException**, **decaf::lang::exceptions::IndexOutOfBoundsException**, **decaf::lang::exceptions::NullPointerException** )

This method transfers long longs long longo this buffer from the given source array.

If there are more long longs to be copied from the array than remain in this buffer, that is, if *length* > **remaining()** (p. 892), then no long longs are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies *length* bytes from the given array long longo this buffer, starting at the given *offset* in the array and at the current position of this buffer. The position of this buffer is then incremented by *length*.

**Parameters**

<i>buffer</i>	The array from which long longs are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of long longs to be read from the given array.

**Returns**

a reference to this buffer.

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only
<i>NullPolong</i>	longerException if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <i>size</i> , <i>offset</i> , or <i>length</i> are not met.

6.498.3.19 **LongBuffer& decaf::nio::LongBuffer::put** ( `std::vector< long long > & buffer` )  
**throw** ( `BufferOverflowException, ReadOnlyBufferException` )

This method transfers the entire content of the given source long longs array long longo this buffer.

This is the same as calling `put( &buffer[0], 0, buffer.size()`.

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>LongBuffer</b> (p. 2403).
---------------	---

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

6.498.3.20 **LongBuffer& decaf::nio::LongBuffer::put** ( `LongBuffer & src` )  
**throw** ( `BufferOverflowException, ReadOnlyBufferException,`  
`lang::exceptions::IllegalArgumentException` )

This method transfers the long longs remaining in the given source buffer long longo this buffer.

If there are more long longs remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 892), then no long longs are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `n = src.remaining()` long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<i>src</i>	The buffer to take long longs from an place in this one.
------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer for the remaining long longs in the source buffer
--	---

<i>IllegalArgumentException</i>	if the source buffer is this buffer
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only

6.498.3.21 `virtual LongBuffer* decaf::nio::LongBuffer::slice ( ) const` [pure virtual]

Creates a new **LongBuffer** (p. 2403) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **LongBuffer** (p. 2403) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 2403).

6.498.3.22 `virtual std::string decaf::nio::LongBuffer::toString ( ) const` [virtual]

#### Returns

a std::string describing this object

6.498.3.23 `static LongBuffer* decaf::nio::LongBuffer::wrap ( long long * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [static]

Wraps the passed buffer with a new **LongBuffer** (p. 2403).

The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

**Returns**

a new **LongBuffer** (p. 2403) that is backed by buffer, caller owns.

**Exceptions**

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.498.3.24 **static LongBuffer\* decaf::nio::LongBuffer::wrap ( std::vector< long long > & buffer ) [static]**

Wraps the passed STL long long Vector in a **LongBuffer** (p. 2403).

The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

**Parameters**

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new **LongBuffer** (p. 2403) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**LongBuffer.h**

**6.499 activemq::util::LongSequenceGenerator Class Reference**

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

```
#include <src/main/activemq/util/LongSequenceGenerator.h>
```

**Public Member Functions**

- **LongSequenceGenerator ()**
- virtual **~LongSequenceGenerator ()**
- long long **getNextSequenceId ()**
- long long **getLastSequenceId ()**

### 6.499.1 Detailed Description

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

This class is thread safe so the ids can be requested in different threads safely.

### 6.499.2 Constructor & Destructor Documentation

6.499.2.1 `activemq::util::LongSequenceGenerator::LongSequenceGenerator ( )`

6.499.2.2 `virtual activemq::util::LongSequenceGenerator::~~LongSequenceGenerator ( )`  
[inline, virtual]

### 6.499.3 Member Function Documentation

6.499.3.1 `long long activemq::util::LongSequenceGenerator::getLastSequenceId ( )`

#### Returns

the last id that was generated.

6.499.3.2 `long long activemq::util::LongSequenceGenerator::getNextSequenceId ( )`

#### Returns

the next id in the sequence.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/LongSequenceGenerator.h`

## 6.500 decaf::net::MalformedURLException Class Reference

```
#include <src/main/decaf/net/MalformedURLException.h>
```

Inheritance diagram for decaf::net::MalformedURLException:

### Public Member Functions

- **MalformedURLException** ( ) throw ( )  
*Default Constructor.*
- **MalformedURLException** (const Exception &ex) throw ( )  
*Conversion Constructor from some other Exception.*

- **MalformedURLException** (const **MalformedURLException** &ex) throw ()  
*Copy Constructor.*
- **MalformedURLException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **MalformedURLException** (const std::exception \*cause) throw ()  
*Constructor.*
- **MalformedURLException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **MalformedURLException** \* **clone** () const  
*Clones this exception.*
- virtual ~**MalformedURLException** () throw ()

### 6.500.1 Constructor & Destructor Documentation

6.500.1.1 `decaf::net::MalformedURLException::MalformedURLException ( ) throw ()`  
[inline]

Default Constructor.

6.500.1.2 `decaf::net::MalformedURLException::MalformedURLException ( const Exception & ex ) throw ()` [inline]

Conversion Constructor from some other Exception.

#### Parameters

ex	An exception that should become this type of Exception
----	--

6.500.1.3 `decaf::net::MalformedURLException::MalformedURLException ( const MalformedURLException & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

ex	An exception that should become this type of Exception
----	--

6.500.1.4 `decaf::net::MalformedURLException::MalformedURLException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.



Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.500.1.5** `decaf::net::MalformedURLException::MalformedURLException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.500.1.6** `decaf::net::MalformedURLException::MalformedURLException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.500.1.7** `virtual decaf::net::MalformedURLException::~~MalformedURLException ( ) throw () [inline, virtual]`

**6.500.2 Member Function Documentation**

**6.500.2.1** `virtual MalformedURLException* decaf::net::MalformedURLException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/MalformedURLException.h`

## 6.501 **decaf::util::Map< K, V, COMPARATOR >** Class Template Reference

**Map** (p. 2419) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

```
#include <src/main/decaf/util/Map.h>
```

Inheritance diagram for `decaf::util::Map< K, V, COMPARATOR >`:

**Data Structures**

- class **Entry**

**Public Member Functions**

- **Map** ()  
*Default constructor - does nothing.*
- virtual **~Map** ()
- virtual bool **equals** (const **Map** &source) const =0  
*Comparison, equality is dependent on the method of determining if the element are equal.*
- virtual void **copy** (const **Map** &source)=0  
*Copies the content of the source map into this map.*
- virtual void **clear** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*
- virtual bool **containsKey** (const K &key) const =0  
*Indicates whether or this map contains a value for the given key.*
- virtual bool **containsValue** (const V &value) const =0  
*Indicates whether or this map contains a value for the given value, i.e.*
- virtual bool **isEmpty** () const =0
- virtual std::size\_t **size** () const =0

- virtual V & **get** (const K &key)=0 throw ( lang::exceptions::NoSuchElementException )  
*Gets the value mapped to the specified key in the **Map** (p. 2419).*
- virtual const V & **get** (const K &key) const =0 throw ( lang::exceptions::NoSuchElementException )  
*Gets the value mapped to the specified key in the **Map** (p. 2419).*
- virtual void **put** (const K &key, const V &value)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Sets the value for the specified key.*
- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Stores a copy of the Mappings contained in the other **Map** (p. 2419) in this one.*
- virtual V **remove** (const K &key)=0 throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.*
- virtual std::vector< K > **keySet** () const =0  
*Returns a **Set** (p. 3379) view of the mappings contained in this map.*
- virtual std::vector< V > **values** () const =0

### 6.501.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>>class decaf::util::Map< K, V, COMPARATOR >

**Map** (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

### 6.501.2 Constructor & Destructor Documentation

6.501.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 decaf::util::Map< K, V, COMPARATOR >::Map ( ) [inline]

Default constructor - does nothing.

6.501.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>>  
 virtual decaf::util::Map< K, V, COMPARATOR >::~~Map ( ) [inline,  
 virtual]

### 6.501.3 Member Function Documentation

```
6.501.3.1  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::Map< K, V, COMPARATOR >::clear ( ) throw (
            decaf::lang::exceptions::UnsupportedOperationException ) [pure
            virtual]
```

Removes all keys and values from this map.

### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1208), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3548), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1208), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3548), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3548), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3548), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3548), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3548), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3548), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3548), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3548), and **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3548).

```
6.501.3.2  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::Map< K, V, COMPARATOR >::containsKey ( const K & key
            ) const [pure virtual]
```

Indicates whether or this map contains a value for the given key.

### Parameters

<i>key</i>	The key to look up.
------------	---------------------

### Returns

true if this map contains the value, otherwise false.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1208), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3548), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1208), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3548), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3548), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3548), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3548), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3548), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3548), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3548), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3548), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3548), and **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3548).

```
6.501.3.3  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::Map< K, V, COMPARATOR >::containsValue ( const V &
            value ) const [pure virtual]
```

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

#### Parameters

<i>value</i>	The Value to look up.
--------------	-----------------------

#### Returns

true if this map contains the value, otherwise false.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1208), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3548), **decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1208), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >**

> (p. 1208), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1208), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3548), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3548), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3548), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3548), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3548), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3548), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3548), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3548), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3548), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3548), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3548), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3548), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3548).

```
6.501.3.4  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::Map< K, V, COMPARATOR >::copy ( const Map< K, V,
                                COMPARATOR > & source ) [pure virtual]
```

Copies the content of the source map into this map.

Erases all existing data in this map.

#### Parameters

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1209), and `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3549).

```
6.501.3.5  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual bool decaf::util::Map< K, V, COMPARATOR >::equals ( const Map< K, V,
                                COMPARATOR > & source ) const [pure virtual]
```

Comparison, equality is dependent on the method of determining if the element are equal.

#### Parameters

<i>source</i>	- <b>Map</b> (p. 2419) to compare to this one.
---------------	--

#### Returns

true if the **Map** (p. 2419) passed is equal in value to this one.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1209), and `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3549).

```
6.501.3.6  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual V& decaf::util::Map< K, V, COMPARATOR >::get ( const K & key ) throw (
           lang::exceptions::NoSuchElementException ) [pure virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2419).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

#### Parameters

<i>key</i>	The search key.
------------	-----------------

#### Returns

A reference to the value for the given key.

#### Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2419).
-------------------------------	--

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1210), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3549), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1210), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3549), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3549), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3549), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3549), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3549), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3549), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3549), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3549), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3549), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3549), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3549), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3549), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3549).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`.

```
6.501.3.7  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual const V& decaf::util::Map< K, V, COMPARATOR >::get ( const K & key )
            const throw ( lang::exceptions::NoSuchElementException ) [pure
            virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2419).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

#### Parameters

<i>key</i>	The search key.
------------	-----------------

#### Returns

A {const} reference to the value for the given key.

#### Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2419).
-------------------------------	--

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1210), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3550), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1210), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1210), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3550), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3550), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3550), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3550), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3550), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3550), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3550), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3550), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3550), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3550), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3550), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3550), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3550).



```
6.501.3.8  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual bool decaf::util::Map< K, V, COMPARATOR >::isEmpty ( ) const
           [pure virtual]
```

### Returns

if the **Map** (p. 2419) contains any element or not, TRUE or FALSE

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1211), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3550), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1211), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1211), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1211), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1211), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1211), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1211), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3550), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3550), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3550), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3550), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3550), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3550), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3550), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3550), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3550), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3550), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3550), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3550), and **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3550).

```
6.501.3.9  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual std::vector<K> decaf::util::Map< K, V, COMPARATOR >::keySet ( )
           const [pure virtual]
```

Returns a **Set** (p. 3379) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2115), **Set.remove** (p. 156), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

### Returns

the entire set of keys in this map as a std::vector.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1211), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3551), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1211), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1211), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1211), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1211), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1211), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1211), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3551), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3551), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3551), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3551), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3551), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3551), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3551), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3551), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3551).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`.

```
6.501.3.10  template<typename K, typename V, typename COMPARATOR
              = std::less<K>> virtual void decaf::util::Map< K, V,
              COMPARATOR >::put ( const K & key, const V & value ) throw (
              decaf::lang::exceptions::UnsupportedOperationException ) [pure
              virtual]
```

Sets the value for the specified key.

#### Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

#### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1212), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3552), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1212), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer<`

ConnectionState >, ConnectionId::COMPARATOR > (p. 1212), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1212), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1212), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1212), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1212), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3552), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3552), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3552), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3552), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3552), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3552), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3552), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3552), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3552), decaf::util::StlMap< int, Pointer< Command > > (p. 3552), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3552), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3552), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3552).

```

6.501.3.11  template<typename K, typename V, typename COMPARATOR =
              std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR
              >::putAll ( const Map< K, V, COMPARATOR > & other ) throw (
              decaf::lang::exceptions::UnsupportedOperationException ) [pure
              virtual]
  
```

Stores a copy of the Mappings contained in the other **Map** (p. 2419) in this one.

#### Parameters

<i>other</i>	A <b>Map</b> (p. 2419) instance whose elements are to all be inserted in this <b>Map</b> (p. 2419).
--------------	---

#### Exceptions

<i>UnsupportedOperationException</i>	If the implementing class does not support the putAll operation.
--------------------------------------	--

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1213), decaf::util::StlMap< K, V, COMPARATOR > (p. 3553), decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR > (p. 1213), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1213), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1213), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1213), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1213), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1213), decaf::util::StlMap<

`cms::Session *`, `SessionResolver *` > (p. 3553), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3553), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3553), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3553), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3553), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3553), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3553), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3553), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3553), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3553), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3553), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3553), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3553).

```
6.501.3.12  template<typename K, typename V, typename COMPARTOR = std::less<K>>
              virtual V decaf::util::Map< K, V, COMPARTOR >::remove ( const K &
              key ) throw ( decaf::lang::exceptions::NoSuchElementException,
              decaf::lang::exceptions::UnsupportedOperationException ) [pure
              virtual]
```

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

#### Parameters

<i>key</i>	The search key.
------------	-----------------

#### Returns

a copy of the element that was previously mapped to the given key

#### Exceptions

<i>NoSuchElementException</i>	if this key is not in the <b>Map</b> (p. 2419).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARTOR >` (p. 1215), `decaf::util::StlMap< K, V, COMPARTOR >` (p. 3553), `decaf::util::concurrent::ConcurrentStlMap< Pointer< Messageld >, Pointer< Message >, Messageld::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1215), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1215), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3553), `decaf::util::StlMap< std::string,`

WireFormatFactory \* > (p. 3553), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3553), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3553), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3553), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3553), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3553), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3553), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3553), decaf::util::StlMap< int, Pointer< Command > > (p. 3553), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3553), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3553), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3553).

```
6.501.3.13  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual std::size_t decaf::util::Map< K, V, COMPARATOR >::size ( ) const
            [pure virtual]
```

## Returns

The number of elements (key/value pairs) in this map.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1217), decaf::util::StlMap< K, V, COMPARATOR > (p. 3554), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1217), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1217), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1217), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1217), decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1217), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1217), decaf::util::StlMap< cms::Session \*, SessionResolver \* > (p. 3554), decaf::util::StlMap< std::string, WireFormatFactory \* > (p. 3554), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 3554), decaf::util::StlMap< std::string, cms::Queue \* > (p. 3554), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3554), decaf::util::StlMap< std::string, CachedConsumer \* > (p. 3554), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3554), decaf::util::StlMap< std::string, TransportFactory \* > (p. 3554), decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3554), decaf::util::StlMap< int, Pointer< Command > > (p. 3554), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3554), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3554), and decaf::util::StlMap< std::string, cms::Topic \* > (p. 3554).

```
6.501.3.14  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual std::vector<V> decaf::util::Map< K, V, COMPARATOR >::values ( )
             const [pure virtual]
```

### Returns

the entire set of values in this map as a `std::vector`.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1218), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3554), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1218), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1218), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1218), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1218), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1218), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1218), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3554), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3554), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3554), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3554), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3554), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3554), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3554), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3554), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3554), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3554), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3554), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3554), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3554).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

## 6.502 cms::MapMessage Class Reference

A **MapMessage** (p. 2431) object is used to send a set of name-value pairs.

```
#include <src/main/cms/MapMessage.h>
```

Inheritance diagram for `cms::MapMessage`:

### Public Member Functions

- virtual `~MapMessage()`

- virtual std::vector< std::string > **getMapNames** () const =0 throw ( CMSExcption )  
*Returns an Enumeration of all the names in the **MapMessage** (p. 2431) object.*
- virtual bool **itemExists** (const std::string &name) const =0 throw ( CMSExcption )  
*Indicates whether an item exists in this **MapMessage** (p. 2431) object.*
- virtual bool **getBoolean** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSExcption )  
*Returns the Boolean value of the Specified name.*
- virtual void **setBoolean** (const std::string &name, bool value)=0 throw ( cms::MessageNotWriteableException, cms::CMSExcption )  
*Sets a boolean value with the specified name into the Map.*
- virtual unsigned char **getByte** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSExcption )  
*Returns the Byte value of the Specified name.*
- virtual void **setByte** (const std::string &name, unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSExcption )  
*Sets a Byte value with the specified name into the Map.*
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSExcption )  
*Returns the Bytes value of the Specified name.*
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSExcption )  
*Sets a Bytes value with the specified name into the Map.*
- virtual char **getChar** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSExcption )  
*Returns the Char value of the Specified name.*
- virtual void **setChar** (const std::string &name, char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSExcption )  
*Sets a Char value with the specified name into the Map.*
- virtual double **getDouble** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSExcption )  
*Returns the Double value of the Specified name.*
- virtual void **setDouble** (const std::string &name, double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSExcption )  
*Sets a Double value with the specified name into the Map.*
- virtual float **getFloat** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSExcption )  
*Returns the Float value of the Specified name.*
- virtual void **setFloat** (const std::string &name, float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSExcption )  
*Sets a Float value with the specified name into the Map.*
- virtual int **getInt** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSExcption )

*Returns the Int value of the Specified name.*

- virtual void **setInt** (const std::string &name, int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Int value with the specified name into the Map.*

- virtual long long **getLong** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the Long value of the Specified name.*

- virtual void **setLong** (const std::string &name, long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Long value with the specified name into the Map.*

- virtual short **getShort** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the Short value of the Specified name.*

- virtual void **setShort** (const std::string &name, short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a Short value with the specified name into the Map.*

- virtual std::string **getString** (const std::string &name) const =0 throw ( cms::MessageFormatException, cms::CMSEException )

*Returns the String value of the Specified name.*

- virtual void **setString** (const std::string &name, const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )

*Sets a String value with the specified name into the Map.*

### 6.502.1 Detailed Description

A **MapMessage** (p. 2431) object is used to send a set of name-value pairs.

The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p. 2431) inherits from the **Message** (p. 2493) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p. 2431), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p. 2680) is thrown. To place the **MapMessage** (p. 2431) back into a state where it can be read from and written to, call the `clearBody` method.

**MapMessage** (p. 2431) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1130). The String-to-primitive conversions may throw a **MessageFormatException** (p. 2622) if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									



byte			X	X		X	X			X
short				X		X	X			X
char					X					X
int						X	X			X
long							X			X
float								X	X	X
double									X	X
String		X	X	X		X	X	X	X	X
byte[]										X
-----										

Since

1.0

6.502.2 Constructor & Destructor Documentation

6.502.2.1 virtual cms::MapMessage::~MapMessage ( ) [inline, virtual]

6.502.3 Member Function Documentation

6.502.3.1 virtual bool cms::MapMessage::getBoolean ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSEException ) [pure virtual]

Returns the Boolean value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

Exceptions

CMSEException (p. 1130)	- if the operation fails due to an internal error.
MessageFormatException	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 335).

6.502.3.2 virtual unsigned char cms::MapMessage::getByte ( const std::string & name ) const throw ( cms::MessageFormatException, cms::CMSEException ) [pure virtual]

Returns the Byte value of the Specified name.

Parameters

name	Name of the value to fetch from the map
------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 335).

```
6.502.3.3  virtual std::vector<unsigned char> cms::MapMessage::getBytes ( const std::string &
                                     name ) const throw ( cms::MessageFormatException, cms::CMSEException
                                     ) [pure virtual]
```

Returns the Bytes value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 335).

```
6.502.3.4  virtual char cms::MapMessage::getChar ( const std::string & name ) const throw (
                                     cms::MessageFormatException, cms::CMSEException ) [pure
                                     virtual]
```

Returns the Char value of the Specified name.

**Parameters**

<i>name</i>	name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 336).

```
6.502.3.5 virtual double cms::MapMessage::getDouble ( const std::string & name ) const
throw ( cms::MessageFormatException, cms::CMSException ) [pure
virtual]
```

Returns the Double value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 336).

```
6.502.3.6 virtual float cms::MapMessage::getFloat ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [pure
virtual]
```

Returns the Float value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 337).

```
6.502.3.7 virtual int cms::MapMessage::getInt ( const std::string & name ) const throw (
cms::MessageFormatException, cms::CMSException ) [pure
virtual]
```

Returns the Int value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 337).

```
6.502.3.8 virtual long long cms::MapMessage::getLong ( const std::string & name ) const
            throw ( cms::MessageFormatException, cms::CMSEException ) [pure
                                virtual]
```

Returns the Long value of the Specified name.

**Parameters**

<i>name</i>	Name of the value to fetch from the map
-------------	---

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 337).

```
6.502.3.9 virtual std::vector< std::string > cms::MapMessage::getMapNames ( ) const throw (
            CMSEException ) [pure virtual]
```

Returns an Enumeration of all the names in the **MapMessage** (p. 2431) object.

**Returns**

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 2431)

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
--	--

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 338).

6.502.3.10 virtual short cms::MapMessage::getShort ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]

Returns the Short value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 338).

6.502.3.11 virtual std::string cms::MapMessage::getString ( const std::string & *name* ) const throw ( cms::MessageFormatException, cms::CMSException ) [pure virtual]

Returns the String value of the Specified name.

#### Parameters

<i>name</i>	Name of the value to fetch from the map
-------------	---

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 339).

6.502.3.12 virtual bool cms::MapMessage::itemExists ( const std::string & *name* ) const throw ( CMSException ) [pure virtual]

Indicates whether an item exists in this **MapMessage** (p. 2431) object.

#### Parameters

<i>name</i>	String name of the Object in question
-------------	---------------------------------------

**Returns**

boolean value indicating if the name is in the map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
--	--

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 339).

6.502.3.13 `virtual void cms::MapMessage::setBoolean ( const std::string & name, bool value )  
throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[pure virtual]`

Sets a boolean value with the specified name into the Map.

**Parameters**

<i>name</i>	the name of the boolean
<i>value</i>	the boolean value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b>	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 340).

6.502.3.14 `virtual void cms::MapMessage::setByte ( const std::string & name,  
unsigned char value ) throw ( cms::MessageNotWriteableException,  
cms::CMSEException ) [pure virtual]`

Sets a Byte value with the specified name into the Map.

**Parameters**

<i>name</i>	the name of the Byte
<i>value</i>	the Byte value to set in the Map

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 340).

```
6.502.3.15  virtual void cms::MapMessage::setBytes ( const std::string &
              name, const std::vector< unsigned char > & value ) throw (
              cms::MessageNotWriteableException, cms::CMSException ) [pure
              virtual]
```

Sets a Bytes value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Bytes
<i>value</i>	The Bytes value to set in the Map

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageNotWriteableException</b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 341).

```
6.502.3.16  virtual void cms::MapMessage::setChar ( const std::string & name, char value )
              throw ( cms::MessageNotWriteableException, cms::CMSException )
              [pure virtual]
```

Sets a Char value with the specified name into the Map.

#### Parameters

<i>name</i>	the name of the Char
<i>value</i>	the Char value to set in the Map

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageNotWriteableException</b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 341).

6.502.3.17 `virtual void cms::MapMessage::setDouble ( const std::string & name, double value )  
throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[pure virtual]`

Sets a Double value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Double
<i>value</i>	The Double value to set in the Map

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 341).

6.502.3.18 `virtual void cms::MapMessage::setFloat ( const std::string & name, float value )  
throw ( cms::MessageNotWriteableException, cms::CMSEException )  
[pure virtual]`

Sets a Float value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the Float
<i>value</i>	The Float value to set in the Map

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 342).

6.502.3.19 `virtual void cms::MapMessage::setInt ( const std::string & name, int value ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets a Int value with the specified name into the Map.



**Parameters**

<i>name</i>	The name of the Int
<i>value</i>	The Int value to set in the Map

**Exceptions**

<b>CMSEException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageNotWriteableException</b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 342).

```
6.502.3.20  virtual void cms::MapMessage::setLong ( const std::string & name, long long value
              ) throw ( cms::MessageNotWriteableException, cms::CMSEException )
              [pure virtual]
```

Sets a Long value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Long
<i>value</i>	The Long value to set in the Map

**Exceptions**

<b>CMSEException</b> (p. 1130)	- if the operation fails due to an internal error.
<b>MessageNotWriteableException</b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 343).

```
6.502.3.21  virtual void cms::MapMessage::setShort ( const std::string & name, short value )
              throw ( cms::MessageNotWriteableException, cms::CMSEException )
              [pure virtual]
```

Sets a Short value with the specified name into the Map.

**Parameters**

<i>name</i>	The name of the Short
<i>value</i>	The Short value to set in the Map

**Exceptions**

<b>CMSEException</b> (p. 1130)	- if the operation fails due to an internal error.
-----------------------------------	--

<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.
---	---

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 343).

6.502.3.22 virtual void cms::MapMessage::setString ( const std::string & *name*, const std::string & *value* ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]

Sets a String value with the specified name into the Map.

#### Parameters

<i>name</i>	The name of the String
<i>value</i>	The String value to set in the Map

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the operation fails due to an internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the <b>Message</b> (p. 2493) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 343).

The documentation for this class was generated from the following file:

- src/main/cms/**MapMessage.h**

## 6.503 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

```
#include <src/main/decaf/util/logging/MarkBlockLogger.h>
```

### Public Member Functions

- **MarkBlockLogger** (**Logger** \*logger, const std::string &blockName)  
*Constructor - Marks Block entry.*
- virtual ~**MarkBlockLogger** ()

### 6.503.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks.

Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

### 6.503.2 Constructor & Destructor Documentation

**6.503.2.1** `decaf::util::logging::MarkBlockLogger::MarkBlockLogger ( Logger * logger, const std::string & blockName ) [inline]`

Constructor - Marks Block entry.

#### Parameters

<i>logger</i>	<b>Logger</b> (p. 2345) to use
<i>blockName</i>	Block name

**6.503.2.2** `virtual decaf::util::logging::MarkBlockLogger::~MarkBlockLogger ( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/MarkBlockLogger.h`

## 6.504 activemq::wireformat::MarshalAware Class Reference

```
#include <src/main/activemq/wireformat/MarshalAware.h>
```

Inheritance diagram for activemq::wireformat::MarshalAware:

### Public Member Functions

- virtual `~MarshalAware ()`
- virtual bool `isMarshalAware () const =0`  
*Determine if the class implementing this interface is really wanting to be told about marshaling.*
- virtual void `beforeMarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`  
*Called before marshaling is started to prepare the object to be marshaled.*
- virtual void `afterMarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`  
*Called after marshaling is started to cleanup the object being marshaled.*
- virtual void `beforeUnmarshal (WireFormat *wireFormat)=0 throw ( decaf::io::IOException )`

*Called before unmarshaling is started to prepare the object to be unmarshaled.*

- virtual void **afterUnmarshal** (**WireFormat** \*wireFormat)=0 throw ( decaf::io::IOException )

*Called after unmarshaling is started to cleanup the object being unmarshaled.*

- virtual void **setMarshaledForm** (**WireFormat** \*wireFormat, const std::vector< char > &data)=0

*Called to set the data to this object that will contain the objects marshaled form.*

- virtual std::vector< unsigned char > **getMarshaledForm** (**WireFormat** \*wireFormat)=0

*Called to get the data to this object that will contain the objects marshaled form.*

## 6.504.1 Constructor & Destructor Documentation

- 6.504.1.1 virtual activemq::wireformat::MarshalAware::~~MarshalAware ( ) [inline, virtual]

## 6.504.2 Member Function Documentation

- 6.504.2.1 virtual void activemq::wireformat::MarshalAware::afterMarshal ( **WireFormat** \* wireFormat ) throw ( decaf::io::IOException ) [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

### Parameters

<i>wireFormat</i>	- the wireformat object to control marshaling
-------------------	---

- 6.504.2.2 virtual void activemq::wireformat::MarshalAware::afterUnmarshal ( **WireFormat** \* wireFormat ) throw ( decaf::io::IOException ) [pure virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

- 6.504.2.3 virtual void activemq::wireformat::MarshalAware::beforeMarshal ( **WireFormat** \* wireFormat ) throw ( decaf::io::IOException ) [pure virtual]

Called before marshaling is started to prepare the object to be marshaled.

### Parameters

<i>wireFormat</i>	- the wireformat object to control marshaling
-------------------	---

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 333), and **activemq::commands::ActiveMQTextMessage** (p. 632).

**6.504.2.4** `virtual void activemq::wireformat::MarshalAware::beforeUnmarshal ( WireFormat * wireFormat ) throw ( decaf::io::IOException ) [pure virtual]`

Called before unmarshaling is started to prepare the object to be unmarshaled.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

**6.504.2.5** `virtual std::vector<unsigned char> activemq::wireformat::MarshalAware::getMarshaledForm ( WireFormat * wireFormat ) [pure virtual]`

Called to get the data to this object that will contain the objects marshaled form.

#### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

#### Returns

buffer that holds the objects data.

**6.504.2.6** `virtual bool activemq::wireformat::MarshalAware::isMarshalAware ( ) const [pure virtual]`

Determine if the class implementing this interface is really wanting to be told about marshaling.

Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

#### Returns

true if this class cares about marshaling.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 339), **activemq::commands::BaseDataStructure** (p. 796), **activemq::commands::Message** (p. 2486), and **activemq::commands::WireFormatInfo** (p. 3918).

**6.504.2.7** `virtual void activemq::wireformat::MarshalAware::setMarshaledForm ( WireFormat * wireFormat, const std::vector< char > & data ) [pure virtual]`

Called to set the data to this object that will contain the objects marshaled form.

**Parameters**

<i>wireFormat</i>	- the wireformat object to control unmarshaling
<i>data</i>	- vector of object binary data

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**MarshalAware.h**

## 6.505 activemq::wireformat::openwire::marshal::v6::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h>
```

**Public Member Functions**

- virtual **~MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** \*format)

### 6.505.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol.

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

### 6.505.2 Constructor & Destructor Documentation

6.505.2.1 virtual activemq::wireformat::openwire::marshal::v6::MarshallerFactory::~~MarshallerFactory  
( ) [inline, virtual]

### 6.505.3 Member Function Documentation

6.505.3.1 virtual void activemq::wireformat::openwire::marshal::v6::MarshallerFactory::configure  
( **OpenWireFormat** \* *format* ) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**MarshallerFactory.h**

## 6.506 activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h>
```

### Public Member Functions

- virtual **~MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** \*format)

#### 6.506.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol.

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.506.2 Constructor & Destructor Documentation

6.506.2.1 virtual activemq::wireformat::openwire::marshal::v3::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]

#### 6.506.3 Member Function Documentation

6.506.3.1 virtual void activemq::wireformat::openwire::marshal::v3::MarshallerFactory::configure ( **OpenWireFormat** \* *format* ) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MarshallerFactory.h**

## 6.507 activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h>
```

### Public Member Functions

- virtual **~MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** \*format)

### 6.507.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol.

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

### 6.507.2 Constructor & Destructor Documentation

6.507.2.1 `virtual activemq::wireformat::openwire::marshal::v4::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]`

### 6.507.3 Member Function Documentation

6.507.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::MarshallerFactory::configure ( OpenWireFormat * format ) [virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h`

## 6.508 activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h>
```

### Public Member Functions

- `virtual ~MarshallerFactory ()`
- `virtual void configure (OpenWireFormat *format)`

### 6.508.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol.

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

### 6.508.2 Constructor & Destructor Documentation

6.508.2.1 `virtual activemq::wireformat::openwire::marshal::v5::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]`



### 6.508.3 Member Function Documentation

6.508.3.1 virtual void activemq::wireformat::openwire::marshal::v5::MarshallerFactory::configure ( OpenWireFormat \* *format* ) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MarshallerFactory.h**

## 6.509 activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h>
```

### Public Member Functions

- virtual ~**MarshallerFactory** ()
- virtual void **configure** (OpenWireFormat \*format)

### 6.509.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol.

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

### 6.509.2 Constructor & Destructor Documentation

6.509.2.1 virtual activemq::wireformat::openwire::marshal::v1::MarshallerFactory::~MarshallerFactory ( ) [inline, virtual]

### 6.509.3 Member Function Documentation

6.509.3.1 virtual void activemq::wireformat::openwire::marshal::v1::MarshallerFactory::configure ( OpenWireFormat \* *format* ) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MarshallerFactory.h**

## 6.510 activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h>
```

### Public Member Functions

- virtual **~MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** \*format)

#### 6.510.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol.

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

#### 6.510.2 Constructor & Destructor Documentation

6.510.2.1 virtual **activemq::wireformat::openwire::marshal::v2::MarshallerFactory::~MarshallerFactory** ( ) [*inline*, *virtual*]

#### 6.510.3 Member Function Documentation

6.510.3.1 virtual void **activemq::wireformat::openwire::marshal::v2::MarshallerFactory::configure** ( **OpenWireFormat** \* *format* ) [*virtual*]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MarshallerFactory.h**

## 6.511 activemq::util::MarshallingSupport Class Reference

```
#include <src/main/activemq/util/MarshallingSupport.h>
```

### Public Member Functions

- **MarshallingSupport** ()
- virtual **~MarshallingSupport** ()

## Static Public Member Functions

- static void **writeString** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )  
*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*
- static void **writeString16** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )  
*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*
- static void **writeString32** (decaf::io::DataOutputStream &dataOut, const std::string &value) throw ( decaf::io::IOException )  
*Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.*
- static std::string **readString16** (decaf::io::DataInputStream &dataIn) throw ( decaf::io::IOException )  
*Reads an Openwire encoded string from the provided DataInputStream.*
- static std::string **readString32** (decaf::io::DataInputStream &dataIn) throw ( decaf::io::IOException )  
*Reads an Openwire encoded string from the provided DataInputStream.*
- static std::string **asciiToModifiedUtf8** (const std::string &asciiString) throw ( decaf::io::UTFDataFormatException )  
*Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.*
- static std::string **modifiedUtf8ToAscii** (const std::string modifiedUtf8String) throw ( decaf::io::UTFDataFormatException )  
*Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].*

## 6.511.1 Constructor & Destructor Documentation

6.511.1.1 **activemq::util::MarshallingSupport::MarshallingSupport** ( )

6.511.1.2 **virtual activemq::util::MarshallingSupport::~~MarshallingSupport** ( ) [virtual]

## 6.511.2 Member Function Documentation

6.511.2.1 **static std::string activemq::util::MarshallingSupport::asciiToModifiedUtf8** ( const std::string & *asciiString* ) throw ( decaf::io::UTFDataFormatException )  
 [static]

Given an ASCII String with byte values [0..255] convert the string to a string containing the modified UTF-8 form of that same string.

This allows an ASCII string containing values greater than 127 as well as embedded NULLs to be sent to a Java client.

**Parameters**

<i>asciiString</i>	The ASCII string to encode as Modified UTF-8
--------------------	--

**Returns**

a string containing the Modified UTF-8 encoded form of the provided string.

**Exceptions**

<i>UTFDataFormatException</i>	if the length of the encoded string would exceed the size of an signed integer.
-------------------------------	---

```
6.511.2.2 static std::string activemq::util::MarshallingSupport::modifiedUtf8ToAscii ( const
std::string modifiedUtf8String ) throw ( decaf::io::UTFDataFormatException )
[static]
```

Given a string that contains bytes in the Java Modified UTF-8 format convert that string back into ASCII values from [0..255].

This will handle any string sent from a Java client which contains values within the [0..255] range or has embedded Nulls. Strings that have encoded values greater than 255 will cause an exception to be thrown.

**Parameters**

<i>modifiedUtf8String</i>	The string to convert from Modified UTF-8 to ASCII.
---------------------------	---

**Returns**

the ASCII encoded version of the provided string.

**Exceptions**

<i>UTFDataFormatException</i>	if the provided string contains invalid data or the character values encoded in the string exceed ASCII value 255.
-------------------------------	--

```
6.511.2.3 static std::string activemq::util::MarshallingSupport::readString16 (
decaf::io::DataInputStream & dataIn ) throw ( decaf::io::IOException )
[static]
```

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 16bits.

**Parameters**

<i>dataIn</i>	The DataInputStream to read the String data from.
---------------	---

**Returns**

the String value.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

**6.511.2.4** `static std::string activemq::util::MarshallingSupport::readString32 ( decaf::io::DataInputStream & dataIn ) throw ( decaf::io::IOException )`  
[static]

Reads an Openwire encoded string from the provided DataInputStream.

No string processing is performed by this method, clients that know the data contains UTF-8 encoded content must use one of the utility methods of this class to decode the UTF-8 data.

This version assumes a size prefix of 32bits.

**Parameters**

<i>dataIn</i>	The DataInputStream to read the String data from.
---------------	---

**Returns**

the String value.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

**6.511.2.5** `static void activemq::util::MarshallingSupport::writeString ( decaf::io::DataOutputStream & dataOut, const std::string & value ) throw ( decaf::io::IOException )` [static]

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed.

**Parameters**

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

6.511.2.6 `static void activemq::util::MarshallingSupport::writeString16 (`  
`decaf::io::DataOutputStream & dataOut, const std::string & value ) throw (`  
`decaf::io::IOException ) [static]`

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a short and the string data no Openwire Type tag is appended.

**Parameters**

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

6.511.2.7 `static void activemq::util::MarshallingSupport::writeString32 (`  
`decaf::io::DataOutputStream & dataOut, const std::string & value ) throw (`  
`decaf::io::IOException ) [static]`

Write the string object to the given DataOutputStream as Raw bytes, no string encoding is done on this char values in the string.

User must encode to Modified UTF-8 as needed. This method write out only the size as a int and the string data no Openwire Type tag is appended.

**Parameters**

<i>dataOut</i>	The DataOutputStream to write the String data to.
<i>value</i>	Thre String value to write in Openwire form.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while writing the string.
--------------------	--

The documentation for this class was generated from the following file:

- `src/main/activemq/util/MarshallingSupport.h`

## 6.512 decaf::lang::Math Class Reference

The class **Math** (p. 2455) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

```
#include <src/main/decaf/lang/Math.h>
```

### Public Member Functions

- **Math** ()
- virtual ~**Math** ()

### Static Public Member Functions

- static int **abs** (int value)  
*Returns the absolute value of an int value.*
- static long long **abs** (long long value)  
*Returns the absolute value of an long long value.*
- static float **abs** (float value)  
*Returns the absolute value of a float value.*
- static double **abs** (double value)  
*Returns the absolute value of a double value.*
- static double **sqrt** (double value)  
*Returns the arc cosine of an angle, in the range of 0.0 through pi.*
- static double **pow** (double base, double exp)  
*Returns the value of the first argument raised to the power of the second argument.*
- static short **min** (short a, short b)  
*Returns the double value that is closest in value to the argument and is equal to a mathematical integer.*
- static int **min** (int a, int b)  
*Returns the smaller of two int values.*
- static unsigned int **min** (unsigned int a, unsigned int b)  
*Returns the smaller of two unsigned int values.*
- static long long **min** (long long a, long long b)  
*Returns the smaller of two long long values.*
- static float **min** (float a, float b)  
*Returns the smaller of two float values.*
- static double **min** (double a, double b)  
*Returns the smaller of two double values.*
- static short **max** (short a, short b)  
*Returns the larger of two short values.*
- static int **max** (int a, int b)  
*Returns the larger of two int values.*
- static long long **max** (long long a, long long b)

*Returns the larger of two `long long` values.*

- static float **max** (float a, float b)

*Returns the greater of two float values.*

- static double **max** (double a, double b)

*Returns the greater of two double values.*

- static double **ceil** (double value)

*Returns the natural logarithm (base e) of a double value.*

- static double **floor** (double value)

*Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.*

- static int **round** (float value)

*Returns the closest int to the argument.*

- static long long **round** (double value)

*Returns the closest long long to the argument.*

- static double **random** ()

*Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.*

- static float **signum** (float value)

*Returns Euler's number e raised to the power of a double value.*

- static double **signum** (double value)

*Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.*

- static double **toRadians** (double angdeg)

*Returns the measure in radians of the supplied degree angle.*

- static double **toDegrees** (double angrad)

*Returns the measure in degrees of the supplied radian angle.*

## Static Public Attributes

- static const double **E**
- static const double **PI**

### 6.512.1 Detailed Description

The class **Math** (p. 2455) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

### 6.512.2 Constructor & Destructor Documentation

6.512.2.1 `decaf::lang::Math::Math ( )` [inline]

6.512.2.2 `virtual decaf::lang::Math::~~Math ( )` [inline, virtual]



### 6.512.3 Member Function Documentation

6.512.3.1 `static int decaf::lang::Math::abs ( int value ) [inline, static]`

Returns the absolute value of an int value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

#### Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

#### Returns

the value if positive, otherwise the negative of value

6.512.3.2 `static long long decaf::lang::Math::abs ( long long value ) [inline, static]`

Returns the absolute value of an long long value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

#### Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

#### Returns

the value if positive, otherwise the negative of value

6.512.3.3 `static double decaf::lang::Math::abs ( double value ) [static]`

Returns the absolute value of a double value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Double::longBitsToDouble** (p. 1757)( 0x7fffffffffffffffULL & Double::doubleToLongBits( *value* ) )

#### Parameters

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

**Returns**

the value if positive, otherwise the negative of value

**6.512.3.4 static float decaf::lang::Math::abs ( float *value* ) [static]**

Returns the absolute value of a float value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Float::intBitsToFloat** (p. 1870)( 0x7fffffff & Float::floatToIntBits( value ) )

**Parameters**

<i>value</i>	- the value to return the abs of
--------------	----------------------------------

**Returns**

the value if positive, otherwise the negative of value

**6.512.3.5 static double decaf::lang::Math::ceil ( double *value* ) [static]**

Returns the natural logarithm (base e) of a double value.

Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

**Parameters**

<i>value</i>	the value to compute the natural log of.
--------------	--

**Returns**

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to 10<sup>n</sup> for integer n, then the result is n.

**Parameters**

<i>value</i>	- the value to operate on
--------------	---------------------------

### Returns

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values  $x$ , the result of  $\log_{10}(x)$  is much closer to the true result of  $\ln(1 + x)$  than the floating-point evaluation of  $\log(1.0+x)$ .

Special cases:

- o If the argument is NaN or less than -1, then the result is NaN.
- o If the argument is positive infinity, then the result is positive infinity.
- o If the argument is negative one, then the result is negative infinity.
- o If the argument is zero, then the result is a zero with the same sign as the argument.

### Parameters

<i>value</i>	- the value to operate on
--------------	---------------------------

### Returns

the the value  $\ln(x + 1)$ , the natural log of  $x + 1$  Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.
- o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of  $\text{Math.ceil}(x)$  is exactly the value of  $-\text{Math.floor}(-x)$ .

### Parameters

<i>value</i>	- the value to find the ceiling of
--------------	------------------------------------

### Returns

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

#### 6.512.3.6 static double decaf::lang::Math::floor ( double *value* ) [static]

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

### Parameters

<i>value</i>	- the value to find the floor of
--------------	----------------------------------

**Returns**

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

**6.512.3.7** `static float decaf::lang::Math::max ( float a, float b )` `[static]`

Returns the greater of two float values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.8** `static double decaf::lang::Math::max ( double a, double b )` `[static]`

Returns the greater of two double values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.9** `static short decaf::lang::Math::max ( short a, short b )` `[inline, static]`

Returns the larger of two `short` values.

That is, the result the argument closer to the value of `Short : :MAX_VALUE` (p.3389). If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.10** `static int decaf::lang::Math::max ( int a, int b ) [inline, static]`

Returns the larger of two `int` values.

That is, the result the argument closer to the value of `Integer::MAX_VALUE` (p. 2054).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.11** `static long long decaf::lang::Math::max ( long long a, long long b ) [inline, static]`

Returns the larger of two `long long` values.

That is, the result the argument closer to the value of `Long::MAX_VALUE` (p. 2392).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the larger of *a* and *b*.

**6.512.3.12** `static int decaf::lang::Math::min ( int a, int b ) [inline, static]`

Returns the smaller of two `int` values.

That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p. 2054).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.13** `static unsigned int decaf::lang::Math::min ( unsigned int a, unsigned int b )`  
`[inline, static]`

Returns the smaller of two `unsigned int` values.

That is, the result the argument closer to the value of **Integer::MIN\_VALUE** (p. 2054).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.14** `static long long decaf::lang::Math::min ( long long a, long long b )` `[inline, static]`

Returns the smaller of two `long long` values.

That is, the result the argument closer to the value of **Long::MIN\_VALUE** (p. 2392).  
If the arguments have the same value, the result is that same value.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.15** `static float decaf::lang::Math::min ( float a, float b )` `[static]`

Returns the smaller of two `float` values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly

smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.16** `static double decaf::lang::Math::min ( double a, double b )` `[static]`

Returns the smaller of two double values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

**Parameters**

<i>a</i>	- an argument.
<i>b</i>	- another argument.

**Returns**

the smaller of *a* and *b*.

**6.512.3.17** `static short decaf::lang::Math::min ( short a, short b )` `[inline, static]`

Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

**Parameters**

<i>value</i>	- the value to round to the nearest integer
--------------	---

**Returns**

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of `decaf::lang::Short::MIN_VALUE` (p. 3389).

If the arguments have the same value, the result is that same value.

#### Parameters

<i>a</i>	- an argument.
<i>b</i>	- another argument.

#### Returns

the smaller of *a* and *b*.

#### 6.512.3.18 static double decaf::lang::Math::pow ( double *base*, double *exp* ) [static]

Returns the value of the first argument raised to the power of the second argument.

Special cases:

o If the second argument is positive or negative zero, then the result is 1.0. o If the second argument is 1.0, then the result is the same as the first argument. o If the second argument is NaN, then the result is NaN. o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

#### Parameters

<i>base</i>	- the base
<i>exp</i>	- the exponent

#### Returns

the base raised to the power of *exp*.

#### 6.512.3.19 static double decaf::lang::Math::random ( ) [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

The remainder value is mathematically equal to  $f1 - f2 \times n$ , where *n* is the mathematical integer closest to the exact mathematical value of the quotient  $f1/f2$ , and if two mathematical integers are equally close to  $f1/f2$ , then *n* is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN. o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

#### Parameters

<i>f1</i>	- the dividend.
<i>f2</i>	- the divisor



**Returns**

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

**Returns**

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

**6.512.3.20 static long long decaf::lang::Math::round ( double value ) [static]**

Returns the closest long long to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long)**Math.floor** (p. 2460)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN\_VALUE** (p. 2392), the result is equal to the value of **Long::MIN\_VALUE** (p. 2392). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX\_VALUE** (p. 2392), the result is equal to the value of **Long::MAX\_VALUE** (p. 2392).

**Parameters**

<i>value</i>	- the value to round
--------------	----------------------

**Returns**

the value of the argument rounded to the nearest integral value.

**6.512.3.21 static int decaf::lang::Math::round ( float value ) [static]**

Returns the closest int to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: (int)**Math.floor** (p. 2460)( a + 0.5f )

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN\_VALUE** (p. 2054), the result is equal to the value of **Integer::MIN\_VALUE** (p. 2054). o If the argument is positive infinity or any

value greater than or equal to the value of **Integer::MAX\_VALUE** (p. 2054), the result is equal to the value of **Integer::MAX\_VALUE** (p. 2054).

#### Parameters

<i>value</i>	- the value to round
--------------	----------------------

#### Returns

the value of the argument rounded to the nearest integral value.

#### 6.512.3.22 static double decaf::lang::Math::signum ( double *value* ) [static]

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

#### Parameters

<i>value</i>	- the floating-point value whose signum is to be returned
--------------	---

#### Returns

the signum function of the argument

#### 6.512.3.23 static float decaf::lang::Math::signum ( float *value* ) [static]

Returns Euler's number  $e$  raised to the power of a double value.

Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

#### Parameters

<i>value</i>	- the exponent to raise $e$ to
--------------	--------------------------------

#### Returns

the value  $e^a$ , where  $e$  is the base of the natural logarithms. Returns  $e^x - 1$ . Note that for values of  $x$  near 0, the exact sum of  $\expm1(x) + 1$  is much closer to the true result of  $e^x$  than  $\exp(x)$ . Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the value to raise $e^x - 1$
--------------	--------------------------------

**Returns**

the value  $e^x - 1$ . Returns  $\sqrt{x^2 + y^2}$  without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

**Parameters**

<i>x</i>	- an argument
<i>y</i>	- another argument

**Returns**

the  $\sqrt{x^2 + y^2}$  without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

**Parameters**

<i>value</i>	- the floating-point value whose signum is to be returned
--------------	---

**Returns**

the signum function of the argument

**6.512.3.24 static double decaf::lang::Math::sqrt ( double *value* ) [static]**

Returns the arc cosine of an angle, in the range of 0.0 through pi.

Special case:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

**Parameters**

<i>value</i>	- the value to return the arc cosine of.
--------------	--

**Returns**

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.  
o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the value to return the arc cosine of.
--------------	--

**Returns**

arc cosine of *value* in radians. Returns the arc tangent of an angle, in the range of  $-\pi/2$  through  $\pi/2$ . Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

**Parameters**

<i>value</i>	- the value to return the arc cosine of.
--------------	--

**Returns**

arc tangent of *value* in radians. Converts rectangular coordinates (*x*, *y*) to polar (*r*, *theta*). This method computes the phase *theta* by computing an arc tangent of *y/x* in the range of  $-\pi$  to  $\pi$ . Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to  $\pi$ . o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to  $-\pi$ . o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to  $\pi/2$ . o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to  $-\pi/2$ . o If both arguments are positive infinity, then the result is the double value closest to  $\pi/4$ . o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to  $3\pi/4$ . o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to  $-\pi/4$ . o If both arguments are negative infinity, then the result is the double value closest to  $-3\pi/4$ .

**Parameters**

<i>y</i>	- the ordinate coordinate
<i>x</i>	- the abscissa coordinate

**Returns**

the *theta* component of the point (*r*, *theta*) in polar coordinates that corresponds to the point (*x*, *y*) in Cartesian coordinates. Returns the cube root of a double value.

For positive finite  $x$ ,  $\text{cbrt}(-x) == -\text{cbrt}(x)$ ; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

#### Parameters

<i>value</i>	- the double to compute the cube root of
--------------	--

#### Returns

the cube root of *value* Returns the trigonometric cosine of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN.

#### Parameters

<i>value</i>	- an value in radians
--------------	-----------------------

#### Returns

the cosine of the argument. Returns the hyperbolic cosine of a double value. The hyperbolic cosine of  $x$  is defined to be  $(e^x + e^{-x})/2$  where  $e$  is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

#### Parameters

<i>value</i>	- the number whose hyperbolic cosine is to be found
--------------	---

#### Returns

the hyperbolic cosine of *value* Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

#### Parameters

<i>value</i>	- the number whose sin is to be found
--------------	---------------------------------------

#### Returns

the sine of *value* Returns the hyperbolic sine of a double value. The hyperbolic sine of  $x$  is defined to be  $(e^x - e^{-x})/2$  where  $e$  is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then

the result is a zero with the same sign as the argument.

#### Parameters

<i>value</i>	- the number whose hyperbolic sin is to be found
--------------	--

#### Returns

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

#### Parameters

<i>value</i>	- the number whose tangent is to be found
--------------	---

#### Returns

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of  $x$  is defined to be  $(e^x - e^{-x}) / (e^x + e^{-x})$ , in other words,  $\sinh(x) / \cosh(x)$ . Note that the absolute value of the exact  $\tanh$  is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

#### Parameters

<i>value</i>	- the number whose hyperbolic tangent is to be found
--------------	--

#### Returns

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

#### Parameters

<i>value</i>	- the value to find the square root of
<i>the</i>	square root of the argument.

6.512.3.25 `static double decaf::lang::Math::toDegrees ( double angrad )` [`inline`, `static`]

Returns the measure in degrees of the supplied radian angle.

#### Parameters

<i>angrad</i>	- an angle in radians
---------------	-----------------------

#### Returns

the degree measure of the angle.

6.512.3.26 `static double decaf::lang::Math::toRadians ( double angdeg )` [`inline`, `static`]

Returns the measure in radians of the supplied degree angle.

#### Parameters

<i>angdeg</i>	- an angle in degrees
---------------	-----------------------

#### Returns

the radian measure of the angle.

### 6.512.4 Field Documentation

6.512.4.1 `const double decaf::lang::Math::E` [`static`]

6.512.4.2 `const double decaf::lang::Math::PI` [`static`]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Math.h`

## 6.513 activemq::util::MemoryUsage Class Reference

```
#include <src/main/activemq/util/MemoryUsage.h>
```

Inheritance diagram for `activemq::util::MemoryUsage`:

### Public Member Functions

- `MemoryUsage ()`

*Default Constructor.*

- **MemoryUsage** (unsigned long long limit)

*Creates an instance of an **Usage** (p. 3895) monitor with a set limit.*

- virtual **~MemoryUsage** ()
- virtual void **waitForSpace** ()

*Waits forever for more space to be returned to this **Usage** (p. 3895) Manager.*

- virtual void **waitForSpace** (unsigned int timeout)

*Waits for more space to be returned to this **Usage** (p. 3895) Manager, times out when the given time span in milliseconds elapses.*

- virtual void **enqueueUsage** (unsigned long long value)

*Tries to increase the usage by value amount but blocks if this object is currently full.*

- virtual void **increaseUsage** (unsigned long long value)

*Increases the usage by the value amount.*

- virtual void **decreaseUsage** (unsigned long long value)

*Decreases the usage by the value amount.*

- virtual bool **isFull** () const

*Returns true if this **Usage** (p. 3895) instance is full, i.e.*

- unsigned long long **getUsage** () const

*Gets the current usage amount.*

- void **setUsage** (unsigned long long usage)

*Sets the current usage amount.*

- unsigned long long **getLimit** () const

*Gets the current limit amount.*

- void **setLimit** (unsigned long long limit)

*Sets the current limit amount.*

### 6.513.1 Constructor & Destructor Documentation

#### 6.513.1.1 activemq::util::MemoryUsage::MemoryUsage ( )

Default Constructor.

#### 6.513.1.2 activemq::util::MemoryUsage::MemoryUsage ( unsigned long long limit )

Creates an instance of an **Usage** (p. 3895) monitor with a set limit.

#### Parameters

<i>limit</i>	- amount of memory this manager allows.
--------------	---

#### 6.513.1.3 virtual activemq::util::MemoryUsage::~~MemoryUsage ( ) [virtual]



## 6.513.2 Member Function Documentation

6.513.2.1 `virtual void activemq::util::MemoryUsage::decreaseUsage ( unsigned long long value )` `[virtual]`

Decreases the usage by the value amount.

### Parameters

<i>value</i>	Amount of space to return to the pool
--------------	---------------------------------------

Implements **activemq::util::Usage** (p. 3896).

6.513.2.2 `virtual void activemq::util::MemoryUsage::enqueueUsage ( unsigned long long value )` `[inline, virtual]`

Tries to increase the usage by value amount but blocks if this object is currently full.

### Parameters

<i>value</i>	Amount of usage in bytes to add.
--------------	----------------------------------

Implements **activemq::util::Usage** (p. 3896).

6.513.2.3 `unsigned long long activemq::util::MemoryUsage::getLimit ( ) const` `[inline]`

Gets the current limit amount.

### Returns

the amount that can be used before full.

6.513.2.4 `unsigned long long activemq::util::MemoryUsage::getUsage ( ) const` `[inline]`

Gets the current usage amount.

### Returns

the amount of bytes currently used.

6.513.2.5 `virtual void activemq::util::MemoryUsage::increaseUsage ( unsigned long long value )` `[virtual]`

Increases the usage by the value amount.

### Parameters

<i>value</i>	Amount of usage to add.
--------------	-------------------------

Implements **activemq::util::Usage** (p. 3896).

6.513.2.6 `virtual bool activemq::util::MemoryUsage::isFull ( ) const` `[virtual]`

Returns true if this **Usage** (p. 3895) instance is full, i.e.

**Usage** (p. 3895)  $\geq 100\%$

Implements **activemq::util::Usage** (p. 3896).

6.513.2.7 `void activemq::util::MemoryUsage::setLimit ( unsigned long long limit )`  
`[inline]`

Sets the current limit amount.

#### Parameters

<i>limit</i>	- The amount that can be used before full.
--------------	--

6.513.2.8 `void activemq::util::MemoryUsage::setUsage ( unsigned long long usage )`  
`[inline]`

Sets the current usage amount.

#### Parameters

<i>usage</i>	- The amount to tag as used.
--------------	------------------------------

6.513.2.9 `virtual void activemq::util::MemoryUsage::waitForSpace ( )` `[virtual]`

Waits forever for more space to be returned to this **Usage** (p. 3895) Manager.

Implements **activemq::util::Usage** (p. 3897).

6.513.2.10 `virtual void activemq::util::MemoryUsage::waitForSpace ( unsigned int timeout )`  
`[virtual]`

Waits for more space to be returned to this **Usage** (p. 3895) Manager, times out when the given time span in milliseconds elapses.

#### Parameters

<i>timeout</i>	The time to wait for more space.
----------------	----------------------------------

Implements **activemq::util::Usage** (p. 3897).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MemoryUsage.h**

## 6.514 activemq::commands::Message Class Reference

```
#include <src/main/activemq/commands/Message.h>
```

Inheritance diagram for activemq::commands::Message:

### Public Member Functions

- **Message** ()
- virtual **~Message** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **Message \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_ - UNUSED) throw ( decaf::io::IOException )  
*Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.*
- virtual void **afterUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_ - UNUSED) throw ( decaf::io::IOException )  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*
- virtual bool **isMarshalAware** () const  
*Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.*
- virtual void **setAckHandler** (const **Pointer**< **core::ActiveMQAckHandler** > &handler)  
*Sets the Acknowledgment Handler that this **Message** (p. 2475) will use when the Acknowledge method is called.*

- virtual **Pointer**< **core::ActiveMQAckHandler** > **getAckHandler** () const  
*Gets the Acknowledgment Handler that this **Message** (p. 2475) will use when the Acknowledge method is called.*
- void **setConnection** (**core::ActiveMQConnection** \***connection**)  
*Sets the ActiveMQConnection instance that this **Command** (p. 1165) was created from when the session create methods are called to create a **Message** (p. 2475).*
- **core::ActiveMQConnection** \* **getConnection** () const  
*Gets the ActiveMQConnection instance that this **Command** (p. 1165) was created from when the session create methods are called to create a **Message** (p. 2475).*
- virtual unsigned int **getSize** () const  
*Returns the Size of this message in Bytes.*
- virtual bool **isExpired** () const  
*Returns if this message has expired, meaning that its Expiration time has elapsed.*
- virtual void **onSend** ()  
*Allows derived **Message** (p. 2475) classes to perform tasks before a message is sent.*
- **util::PrimitiveMap** & **getMessageProperties** ()  
*Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.*
- const **util::PrimitiveMap** & **getMessageProperties** () const
- bool **isReadOnlyProperties** () const  
*Returns if the **Message** (p. 2475) Properties Are Read Only.*
- void **setReadOnlyProperties** (bool value)  
*Set the Read Only State of the **Message** (p. 2475) Properties.*
- bool **isReadOnlyBody** () const  
*Returns if the **Message** (p. 2475) Body is Read Only.*
- void **setReadOnlyBody** (bool value)  
*Set the Read Only State of the **Message** (p. 2475) Content.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &**producerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** ()
- virtual void **setOriginalDestination** (const **Pointer**< **ActiveMQDestination** > &**originalDestination**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)

- virtual const **Pointer**< **TransactionId** > & **getOriginalTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getOriginalTransactionId** ()
- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &**originalTransactionId**)
- virtual const std::string & **getGroupId** () const
- virtual std::string & **getGroupId** ()
- virtual void **setGroupId** (const std::string &**groupId**)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int **groupSequence**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual bool **isPersistent** () const
- virtual void **setPersistent** (bool **persistent**)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long **expiration**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getReplyTo** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer**< **ActiveMQDestination** > &**replyTo**)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long **timestamp**)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &**type**)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &**content**)
- virtual const std::vector< unsigned char > & **getMarshaledProperties** () const
- virtual std::vector< unsigned char > & **getMarshaledProperties** ()
- virtual void **setMarshaledProperties** (const std::vector< unsigned char > &**marshalledProperties**)
- virtual const **Pointer**< **DataStructure** > & **getDataStructure** () const
- virtual **Pointer**< **DataStructure** > & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer**< **DataStructure** > &**dataStructure**)
- virtual const **Pointer**< **ConsumerId** > & **getTargetConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer**< **ConsumerId** > &**targetConsumerId**)
- virtual bool **isCompressed** () const
- virtual void **setCompressed** (bool **compressed**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const

- virtual std::vector< **decaf::lang::Pointer< BrokerId > >** & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId > >** &**brokerPath**)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long **arrival**)
- virtual const std::string & **getUserID** () const
- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &**userID**)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool **recievedByDFBridge**)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool **droppable**)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId > >** & **getCluster** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId > >** & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer< BrokerId > >** &**cluster**)
- virtual long long **getBrokerInTime** () const
- virtual void **setBrokerInTime** (long long **brokerInTime**)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long **brokerOutTime**)
- virtual bool **isMessage** () const
- virtual **Pointer< Command >** **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGE** = 0

### Protected Attributes

- **core::ActiveMQConnection** \* **connection**
- **Pointer< ProducerId >** **producerId**
- **Pointer< ActiveMQDestination >** **destination**
- **Pointer< TransactionId >** **transactionId**
- **Pointer< ActiveMQDestination >** **originalDestination**
- **Pointer< Messageld >** **messageld**
- **Pointer< TransactionId >** **originalTransactionId**
- std::string **groupId**
- int **groupSequence**
- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**

- **Pointer**< **ActiveMQDestination** > **replyTo**
- long long **timestamp**
- std::string **type**
- std::vector< unsigned char > **content**
- std::vector< unsigned char > **marshalledProperties**
- **Pointer**< **DataStructure** > **dataStructure**
- **Pointer**< **ConsumerId** > **targetConsumerId**
- bool **compressed**
- int **redeliveryCounter**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**
- long long **arrival**
- std::string **userId**
- bool **recievedByDFBridge**
- bool **droppable**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **cluster**
- long long **brokerInTime**
- long long **brokerOutTime**

### Static Protected Attributes

- static const unsigned int **DEFAULT\_MESSAGE\_SIZE** = 1024

## 6.514.1 Constructor & Destructor Documentation

6.514.1.1 `activemq::commands::Message::Message ( )`

6.514.1.2 `virtual activemq::commands::Message::~~Message ( )` [virtual]

## 6.514.2 Member Function Documentation

6.514.2.1 `virtual void activemq::commands::Message::afterUnmarshal ( wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw ( decaf::io::IOException )` [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

### Parameters

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

Reimplemented from `activemq::commands::BaseDataStructure` (p. 794).

```
6.514.2.2 virtual void activemq::commands::Message::beforeMarshal (
    wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw (
    decaf::io::IOException ) [virtual]
```

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

#### Parameters

<i>wireFormat</i>	- the wireformat controller
-------------------	-----------------------------

Reimplemented from **activemq::commands::BaseDataStructure** (p. 794).

```
6.514.2.3 virtual Message* activemq::commands::Message::cloneDataStructure ( ) const
    [virtual]
```

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 174), **activemq::commands::ActiveMQMapMessage** (p. 205), **activemq::commands::ActiveMQMapMessage** (p. 334), **activemq::commands::ActiveMQMessage** (p. 369), **activemq::commands::ActiveMQObjectMessage** (p. 415), **activemq::commands::ActiveMQStreamMessage** (p. 510), and **activemq::commands::ActiveMQTextMessage** (p. 633).

```
6.514.2.4 virtual void activemq::commands::Message::copyDataStructure ( const
    DataStructure * src ) [virtual]
```

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 174), **activemq::commands::ActiveMQMapMessage** (p. 205), **activemq::commands::ActiveMQMapMessage** (p. 334), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 415), **activemq::commands::ActiveMQStreamMessage** (p. 510), and **activemq::commands::ActiveMQTextMessage** (p. 633).



6.514.2.5 `virtual bool activemq::commands::Message::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 175), **activemq::commands::ActiveMQBytesMessage** (p. 206), **activemq::commands::ActiveMQMapMessage** (p. 334), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 399), **activemq::commands::ActiveMQObjectMessage** (p. 415), **activemq::commands::ActiveMQStreamMessage** (p. 510), **activemq::commands::ActiveMQTextMessage** (p. 633), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 399), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 399), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 399).

6.514.2.6 `virtual Pointer<core::ActiveMQAckHandler> activemq::commands::Message::getAckHandler ( ) const [inline, virtual]`

Gets the Acknowledgment Handler that this **Message** (p. 2475) will use when the Acknowledge method is called.

#### Returns

handler ActiveMQAckHandler to call or NULL if not set

6.514.2.7 `virtual long long activemq::commands::Message::getArrival ( ) const [virtual]`

6.514.2.8 `virtual long long activemq::commands::Message::getBrokerInTime ( ) const [virtual]`

6.514.2.9 `virtual long long activemq::commands::Message::getBrokerOutTime ( ) const [virtual]`

6.514.2.10 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath ( ) const [virtual]`

6.514.2.11 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath ( ) [virtual]`

6.514.2.12 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster ( ) [virtual]`

6.514.2.13 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster ( ) const [virtual]`

6.514.2.14 `core::ActiveMQConnection* activemq::commands::Message::getConnection ( ) const [inline]`

Gets the ActiveMQConnection instance that this **Command** (p. 1165) was created from when the session create methods are called to create a **Message** (p. 2475).

### Returns

the ActiveMQConnection parent for this **Message** (p. 2475) or NULL if not set.

6.514.2.15 `virtual const std::vector<unsigned char>& activemq::commands::Message::getContent ( ) const [virtual]`

6.514.2.16 `virtual std::vector<unsigned char>& activemq::commands::Message::getContent ( ) [virtual]`

6.514.2.17 `virtual const std::string& activemq::commands::Message::getCorrelationId ( ) const [virtual]`

6.514.2.18 `virtual std::string& activemq::commands::Message::getCorrelationId ( ) [virtual]`

6.514.2.19 `virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure ( ) [virtual]`

6.514.2.20 `virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure ( ) const [virtual]`

6.514.2.21 `virtual unsigned char activemq::commands::Message::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 175), **activemq::commands::ActiveMQBytesMessage** (p. 207), **activemq::commands::ActiveMQMapMessage** (p. 336), **activemq::commands::ActiveMQMessage** (p. 370), **activemq::commands::ActiveMQObjectMessage** (p. 416), **activemq::commands::ActiveMQStreamMessage** (p. 510), and **activemq::commands::ActiveMQTextMessage** (p. 633).

- 6.514.2.22 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ( ) const [virtual]`
- 6.514.2.23 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination ( ) [virtual]`
- 6.514.2.24 `virtual long long activemq::commands::Message::getExpiration ( ) const [virtual]`
- 6.514.2.25 `virtual const std::string& activemq::commands::Message::getGroupID ( ) const [virtual]`
- 6.514.2.26 `virtual std::string& activemq::commands::Message::getGroupID ( ) [virtual]`
- 6.514.2.27 `virtual int activemq::commands::Message::getGroupSequence ( ) const [virtual]`
- 6.514.2.28 `virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties ( ) const [virtual]`
- 6.514.2.29 `virtual std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties ( ) [virtual]`
- 6.514.2.30 `virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId ( ) const [virtual]`
- 6.514.2.31 `virtual Pointer<MessageId>& activemq::commands::Message::getMessageId ( ) [virtual]`
- 6.514.2.32 `util::PrimitiveMap& activemq::commands::Message::getMessageProperties ( ) [inline]`

Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

### Returns

a reference to the Primitive Map that holds message properties.

- 6.514.2.33 `const util::PrimitiveMap& activemq::commands::Message::getMessageProperties ( ) const [inline]`
- 6.514.2.34 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ( ) [virtual]`
- 6.514.2.35 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination ( ) const [virtual]`
- 6.514.2.36 `virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ( ) const [virtual]`
- 6.514.2.37 `virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId ( ) [virtual]`
- 6.514.2.38 `virtual unsigned char activemq::commands::Message::getPriority ( ) const [virtual]`
- 6.514.2.39 `virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId ( ) const [virtual]`
- 6.514.2.40 `virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId ( ) [virtual]`
- 6.514.2.41 `virtual int activemq::commands::Message::getRedeliveryCounter ( ) const [virtual]`
- 6.514.2.42 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ( ) const [virtual]`
- 6.514.2.43 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo ( ) [virtual]`
- 6.514.2.44 `virtual unsigned int activemq::commands::Message::getSize ( ) const [virtual]`

Returns the Size of this message in Bytes.

### Returns

number of bytes this message equates to.

Reimplemented in **activemq::commands::ActiveMQTextMessage** (p. 634).

- 6.514.2.45 `virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ( ) const`  
[virtual]
- 6.514.2.46 `virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId ( )`  
[virtual]
- 6.514.2.47 `virtual long long activemq::commands::Message::getTimestamp ( ) const`  
[virtual]
- 6.514.2.48 `virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId ( ) const`  
[virtual]
- 6.514.2.49 `virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId ( )`  
[virtual]
- 6.514.2.50 `virtual const std::string& activemq::commands::Message::getType ( ) const`  
[virtual]
- 6.514.2.51 `virtual std::string& activemq::commands::Message::getType ( )` [virtual]
- 6.514.2.52 `virtual const std::string& activemq::commands::Message::getUserID ( ) const`  
[virtual]
- 6.514.2.53 `virtual std::string& activemq::commands::Message::getUserID ( )` [virtual]
- 6.514.2.54 `virtual bool activemq::commands::Message::isCompressed ( ) const`  
[virtual]
- 6.514.2.55 `virtual bool activemq::commands::Message::isDroppable ( ) const` [virtual]
- 6.514.2.56 `virtual bool activemq::commands::Message::isExpired ( ) const` [virtual]

Returns if this message has expired, meaning that its Expiration time has elapsed.

#### Returns

true if message is expired.

- 6.514.2.57 `virtual bool activemq::commands::Message::isMarshalAware ( ) const`  
[inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.



**cms::Message** > (p. 407), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 407), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 407), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 407).

6.514.2.64 **virtual void activemq::commands::Message::setAckHandler ( const Pointer< core::ActiveMQAckHandler > & handler )** [inline, virtual]

Sets the Acknowledgment Handler that this **Message** (p. 2475) will use when the Acknowledge method is called.

#### Parameters

<i>handler</i>	ActiveMQAckHandler to call
----------------	----------------------------

6.514.2.65 **virtual void activemq::commands::Message::setArrival ( long long arrival )** [virtual]

6.514.2.66 **virtual void activemq::commands::Message::setBrokerInTime ( long long brokerInTime )** [virtual]

6.514.2.67 **virtual void activemq::commands::Message::setBrokerOutTime ( long long brokerOutTime )** [virtual]

6.514.2.68 **virtual void activemq::commands::Message::setBrokerPath ( const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath )** [virtual]

6.514.2.69 **virtual void activemq::commands::Message::setCluster ( const std::vector< decaf::lang::Pointer< BrokerId > > & cluster )** [virtual]

6.514.2.70 **virtual void activemq::commands::Message::setCompressed ( bool compressed )** [virtual]

6.514.2.71 **void activemq::commands::Message::setConnection ( core::ActiveMQConnection \* connection )** [inline]

Sets the ActiveMQConnection instance that this **Command** (p. 1165) was created from when the session create methods are called to create a **Message** (p. 2475).

#### Parameters

<i>handler</i>	ActiveMQConnection parent for this message
----------------	--

6.514.2.72 **virtual void activemq::commands::Message::setContent ( const std::vector< unsigned char > & content )** [virtual]

- 6.514.2.73 `virtual void activemq::commands::Message::setCorrelationId ( const std::string & correlationId ) [virtual]`
- 6.514.2.74 `virtual void activemq::commands::Message::setDataStructure ( const Pointer< DataSet > & dataStructure ) [virtual]`
- 6.514.2.75 `virtual void activemq::commands::Message::setDestination ( const Pointer< ActiveMQDestination > & destination ) [virtual]`
- 6.514.2.76 `virtual void activemq::commands::Message::setDroppable ( bool droppable ) [virtual]`
- 6.514.2.77 `virtual void activemq::commands::Message::setExpiration ( long long expiration ) [virtual]`
- 6.514.2.78 `virtual void activemq::commands::Message::setGroupID ( const std::string & groupId ) [virtual]`
- 6.514.2.79 `virtual void activemq::commands::Message::setGroupSequence ( int groupSequence ) [virtual]`
- 6.514.2.80 `virtual void activemq::commands::Message::setMarshaledProperties ( const std::vector< unsigned char > & marshalledProperties ) [virtual]`
- 6.514.2.81 `virtual void activemq::commands::Message::setMessageId ( const Pointer< MessageId > & messageId ) [virtual]`
- 6.514.2.82 `virtual void activemq::commands::Message::setOriginalDestination ( const Pointer< ActiveMQDestination > & originalDestination ) [virtual]`
- 6.514.2.83 `virtual void activemq::commands::Message::setOriginalTransactionId ( const Pointer< TransactionId > & originalTransactionId ) [virtual]`
- 6.514.2.84 `virtual void activemq::commands::Message::setPersistent ( bool persistent ) [virtual]`
- 6.514.2.85 `virtual void activemq::commands::Message::setPriority ( unsigned char priority ) [virtual]`
- 6.514.2.86 `virtual void activemq::commands::Message::setProducerId ( const Pointer< ProducerId > & producerId ) [virtual]`
- 6.514.2.87 `void activemq::commands::Message::setReadOnlyBody ( bool value ) [inline]`

Set the Read Only State of the **Message** (p. 2475) Content.



**Parameters**

<i>value</i>	- true if Content should be read only.
--------------	--

6.514.2.88 `void activemq::commands::Message::setReadOnlyProperties ( bool value )`  
`[inline]`

Set the Read Only State of the **Message** (p. 2475) Properties.

**Parameters**

<i>value</i>	- true if Properties should be read only.
--------------	---

6.514.2.89 `virtual void activemq::commands::Message::setRecievedByDFBridge ( bool recievedByDFBridge )` `[virtual]`

6.514.2.90 `virtual void activemq::commands::Message::setRedeliveryCounter ( int redeliveryCounter )` `[virtual]`

6.514.2.91 `virtual void activemq::commands::Message::setReplyTo ( const Pointer< ActiveMQDestination > & replyTo )` `[virtual]`

6.514.2.92 `virtual void activemq::commands::Message::setTargetConsumerId ( const Pointer< ConsumerId > & targetConsumerId )` `[virtual]`

6.514.2.93 `virtual void activemq::commands::Message::setTimestamp ( long long timestamp )`  
`[virtual]`

6.514.2.94 `virtual void activemq::commands::Message::setTransactionId ( const Pointer< TransactionId > & transactionId )` `[virtual]`

6.514.2.95 `virtual void activemq::commands::Message::setType ( const std::string & type )`  
`[virtual]`

6.514.2.96 `virtual void activemq::commands::Message::setUserID ( const std::string & userId )`  
`[virtual]`

6.514.2.97 `virtual std::string activemq::commands::Message::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 177), `activemq::commands::ActiveMQMapMessage` (p. 214), `activemq::commands::ActiveMQMessage` (p. 370), `activemq::commands::ActiveMQObjectMessage` (p. 416), `activemq::commands::ActiveMQStreamMessage` (p. 518), and `activemq::commands::ActiveMQTextMessage` (p. 635).

**6.514.2.98** `virtual Pointer<Command> activemq::commands::Message::visit  
( activemq::state::CommandVisitor * visitor ) throw (  
exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 1170).

## 6.514.3 Field Documentation

**6.514.3.1** `long long activemq::commands::Message::arrival [protected]`

**6.514.3.2** `long long activemq::commands::Message::brokerInTime  
[protected]`

**6.514.3.3** `long long activemq::commands::Message::brokerOutTime  
[protected]`

**6.514.3.4** `std::vector< decaf::lang::Pointer<BrokerId> >  
activemq::commands::Message::brokerPath [protected]`

**6.514.3.5** `std::vector< decaf::lang::Pointer<BrokerId> >  
activemq::commands::Message::cluster [protected]`

**6.514.3.6** `bool activemq::commands::Message::compressed [protected]`

**6.514.3.7** `core::ActiveMQConnection* ac-  
tivemq::commands::Message::connection  
[protected]`

**6.514.3.8** `std::vector<unsigned char> activemq::commands::Message::content  
[protected]`

**6.514.3.9** `std::string activemq::commands::Message::correlationId  
[protected]`

- 6.514.3.10 **Pointer<DataStructure> activemq::commands::Message::dataStructure**  
[protected]
- 6.514.3.11 **const unsigned int activemq::commands::Message::DEFAULT\_MESSAGE\_SIZE = 1024** [static, protected]
- 6.514.3.12 **Pointer<ActiveMQDestination> activemq::commands::Message::destination**  
[protected]
- 6.514.3.13 **bool activemq::commands::Message::droppable** [protected]
- 6.514.3.14 **long long activemq::commands::Message::expiration** [protected]
- 6.514.3.15 **std::string activemq::commands::Message::groupID** [protected]
- 6.514.3.16 **int activemq::commands::Message::groupSequence** [protected]
- 6.514.3.17 **const unsigned char activemq::commands::Message::ID\_MESSAGE = 0**  
[static]
- 6.514.3.18 **std::vector<unsigned char> activemq::commands::Message::marshalledProperties**  
[protected]
- 6.514.3.19 **Pointer<MessageId> activemq::commands::Message::messageId**  
[protected]
- 6.514.3.20 **Pointer<ActiveMQDestination> activemq::commands::Message::originalDestination**  
[protected]
- 6.514.3.21 **Pointer<TransactionId> activemq::commands::Message::originalTransactionId**  
[protected]
- 6.514.3.22 **bool activemq::commands::Message::persistent** [protected]
- 6.514.3.23 **unsigned char activemq::commands::Message::priority** [protected]
- 6.514.3.24 **Pointer<ProducerId> activemq::commands::Message::producerId**  
[protected]
- 6.514.3.25 **bool activemq::commands::Message::recievedByDFBridge**  
[protected]
- 6.514.3.26 **int activemq::commands::Message::redeliveryCounter**  
[protected]

- 6.514.3.27 **Pointer<ActiveMQDestination> activemq::commands::Message::replyTo** [protected]
- 6.514.3.28 **Pointer<ConsumerId> activemq::commands::Message::targetConsumerId** [protected]
- 6.514.3.29 **long long activemq::commands::Message::timestamp** [protected]
- 6.514.3.30 **Pointer<TransactionId> activemq::commands::Message::transactionId** [protected]
- 6.514.3.31 **std::string activemq::commands::Message::type** [protected]
- 6.514.3.32 **std::string activemq::commands::Message::userId** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Message.h**

## 6.515 cms::Message Class Reference

Root of all messages.

```
#include <src/main/cms/Message.h>
```

Inheritance diagram for cms::Message:

### Public Member Functions

- virtual **~Message** ()
- virtual **Message \* clone** () const =0  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- virtual void **acknowledge** () const =0 throw ( IllegalStateException, CMSException )  
*Acknowledges all consumed messages of the session of this consumed message.*
- virtual void **clearBody** ()=0 throw ( CMSException )  
*Clears out the body of the message.*
- virtual void **clearProperties** ()=0 throw ( CMSException )  
*Clears out the message body.*
- virtual std::vector< std::string > **getPropertyNames** () const =0 throw ( CMSException )  
*Retrieves the property names.*

- virtual bool **propertyExists** (const std::string &name) const =0 throw ( CMSException )  
*Indicates whether or not a given property exists.*
- virtual bool **getBooleanProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a double property.*
- virtual float **getFloatProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a float property.*
- virtual int **getIntProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a int property.*
- virtual long long **getLongProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a long property.*
- virtual short **getShortProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a short property.*
- virtual std::string **getStringProperty** (const std::string &name) const =0 throw ( MessageFormatException, CMSException )  
*Gets a string property.*
- virtual void **setBooleanProperty** (const std::string &name, bool value)=0 throw ( MessageNotWriteableException, CMSException )  
*Sets a boolean property.*
- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0 throw ( MessageNotWriteableException, CMSException )  
*Sets a byte property.*
- virtual void **setDoubleProperty** (const std::string &name, double value)=0 throw ( MessageNotWriteableException, CMSException )  
*Sets a double property.*
- virtual void **setFloatProperty** (const std::string &name, float value)=0 throw ( MessageNotWriteableException, CMSException )  
*Sets a float property.*
- virtual void **setIntProperty** (const std::string &name, int value)=0 throw ( MessageNotWriteableException, CMSException )  
*Sets a int property.*
- virtual void **setLongProperty** (const std::string &name, long long value)=0 throw ( MessageNotWriteableException, CMSException )  
*Sets a long property.*

- virtual void **setShortProperty** (const std::string &name, short value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a short property.*
- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0 throw ( MessageNotWriteableException, CMSEException )  
*Sets a string property.*
- virtual std::string **getCMSCorrelationID** () const =0 throw ( CMSEException )  
*Gets the correlation ID for the message.*
- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0 throw ( CMSEException )  
*Sets the correlation ID for the message.*
- virtual int **getCMSDeliveryMode** () const =0 throw ( CMSEException )  
*Gets the **DeliveryMode** (p. 1687) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0 throw ( CMSEException )  
*Sets the **DeliveryMode** (p. 1687) for this message.*
- virtual const **Destination** \* **getCMSDestination** () const =0 throw ( CMSEException )  
*Gets the **Destination** (p. 1688) object for this message.*
- virtual void **setCMSDestination** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Sets the **Destination** (p. 1688) object for this message.*
- virtual long long **getCMSExpiration** () const =0 throw ( CMSEException )  
*Gets the message's expiration value.*
- virtual void **setCMSExpiration** (long long expireTime)=0 throw ( CMSEException )  
*Sets the message's expiration value.*
- virtual std::string **getCMSMessageID** () const =0 throw ( CMSEException )  
*The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.*
- virtual void **setCMSMessageID** (const std::string &id)=0 throw ( CMSEException )  
*Sets the message ID.*
- virtual int **getCMSPriority** () const =0 throw ( CMSEException )  
*Gets the message priority level.*
- virtual void **setCMSPriority** (int priority)=0 throw ( CMSEException )  
*Sets the Priority Value for this message.*
- virtual bool **getCMSRedelivered** () const =0 throw ( CMSEException )  
*Gets an indication of whether this message is being redelivered.*
- virtual void **setCMSRedelivered** (bool redelivered)=0 throw ( CMSEException )  
*Specifies whether this message is being redelivered.*
- virtual const **cms::Destination** \* **getCMSReplyTo** () const =0 throw ( CMSEException )  
*Gets the **Destination** (p. 1688) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (const **cms::Destination** \*destination)=0 throw ( CMSEException )

Sets the **Destination** (p. 1688) object to which a reply to this message should be sent.

- virtual long long **getCMSTimestamp** () const =0 throw ( CMSEException )

*Gets the message timestamp.*

- virtual void **setCMSTimestamp** (long long timeStamp)=0 throw ( CMSEException )

*Sets the message timestamp.*

- virtual std::string **getCMSType** () const =0 throw ( CMSEException )

*Gets the message type identifier supplied by the client when the message was sent.*

- virtual void **setCMSType** (const std::string &type)=0 throw ( CMSEException )

*Sets the message type.*

### 6.515.1 Detailed Description

Root of all messages.

As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

#### Message (p. 2493) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 2493) Interface definition.

- Stream - A **StreamMessage** (p. 3595) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 1023) type the values written to a **StreamMessage** (p. 3595) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 2493) Body.
- Map - A **MapMessage** (p. 2431) object's message body contains a set of name-value pairs, where names are std::string objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 2431) makes no guarantee on the order of the elements within the **Message** (p. 2493) body.
- Text - A **TextMessage** (p. 3704) object's message body contains a std::string object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 1023) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

#### Message (p. 2493) Properties

**Message** (p. 2493) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 1130). The String-to-primitive conversions may throw a runtime exception if the primitive's valueOf method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X							X
byte		X	X	X	X			X
short			X	X	X			X
int				X	X			X
long					X			X
float						X	X	X
double							X	X
String	X	X	X	X	X	X	X	X

When a **Message** (p. 2493) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered Message's properties will result in a **CMSException** (p. 1130) being thrown.

#### See also

JMS API

#### Since

1.0

### 6.515.2 Constructor & Destructor Documentation

6.515.2.1 `virtual cms::Message::~Message ( ) [inline, virtual]`

### 6.515.3 Member Function Documentation

6.515.3.1 `virtual void cms::Message::acknowledge ( ) const throw ( IllegalStateException, CMSException ) [pure virtual]`

Acknowledges all consumed messages of the session of this consumed message.

All consumed CMS messages support the acknowledge method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking acknowledge on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to acknowledge are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling acknowledge on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

#### Exceptions



<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
<b><i>IllegalStateException</i></b> (p. 1958)	- if this method is called on a closed session.

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 398), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 398).

**6.515.3.2** `virtual void cms::Message::clearBody ( ) throw ( CMSEException ) [pure virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 205), **activemq::commands::ActiveMQMapMessage** (p. 333), **activemq::commands::ActiveMQStreamMessage** (p. 509), **activemq::commands::ActiveMQTextMessage** (p. 632), **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 398), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 398), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 398).

**6.515.3.3** `virtual void cms::Message::clearProperties ( ) throw ( CMSEException ) [pure virtual]`

Clears out the message body.

Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 399), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 399).

6.515.3.4 `virtual Message* cms::Message::clone ( ) const [pure virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

### Returns

new copy of this message

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 174), `activemq::commands::ActiveMQB`  
 (p. 205), `activemq::commands::ActiveMQMapMessage` (p. 334), `activemq::commands::ActiveMQMessage`  
 (p. 369), `activemq::commands::ActiveMQObjectMessage` (p. 415), `activemq::commands::ActiveMQStream`  
 (p. 509), `activemq::commands::ActiveMQTextMessage` (p. 632), and `cms::BytesMessage`  
 (p. 1026).

6.515.3.5 `virtual bool cms::Message::getBooleanProperty ( const std::string & name ) const`  
`throw ( MessageFormatException, CMSException ) [pure virtual]`

Gets a boolean property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

### Returns

The value for the named property.

### Exceptions

<b>CMSException</b> (p. 1130)	if the property does not exist.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 399), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 399), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`

> (p. 399).

6.515.3.6 virtual unsigned char cms::Message::getBytesProperty ( const std::string & name )  
const throw ( MessageFormatException, CMSEException ) [pure  
virtual]

Gets a byte property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b>CMSEException</b> (p. 1130)	if the property does not exist.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 400), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 400).

6.515.3.7 virtual std::string cms::Message::getCMSCorrelationID ( ) const throw ( CMSEException ) [pure virtual]

Gets the correlation ID for the message.

This method is used to return correlation ID values that are either provider-specific message IDs or application-specific String values.

#### Returns

string representation of the correlation Id

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`

> (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`  
 (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 400), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 400), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 400).

6.515.3.8 `virtual int cms::Message::getCMSDeliveryMode ( ) const throw ( CMSEException )`  
`[pure virtual]`

Gets the **DeliveryMode** (p. 1687) for this message.

#### Returns

**DeliveryMode** (p. 1687) enumerated value.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`  
 (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 401), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 401).

6.515.3.9 `virtual const Destination* cms::Message::getCMSDestination ( ) const throw (`  
`CMSEException ) [pure virtual]`

Gets the **Destination** (p. 1688) object for this message.

The CMSDestination header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its CMSDestination value must be equivalent to the value assigned when it was sent.

#### Returns

**Destination** (p. 1688) object

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 401), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 401).

6.515.3.10 `virtual long long cms::Message::getCMSExpiration ( ) const throw (`  
`CMSException ) [pure virtual]`

Gets the message's expiration value.

When a message is sent, the CMSExpiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, CMSExpiration is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

### Returns

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 401), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 401), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 401).

6.515.3.11 `virtual std::string cms::Message::getCMSMessageID ( ) const throw (`  
`CMSException ) [pure virtual]`

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.

When a message is sent, `CMSMessageID` can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A `CMSMessageID` is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All `CMSMessageID` values must start with the prefix 'ID:'. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used by an application. By calling the **`MessageProducer.setDisableMessageID`** (p. 2688) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

### Returns

provider-assigned message id

### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 402), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 402).

**6.515.3.12** `virtual int cms::Message::getCMSPriority ( ) const throw ( CMSException )`  
[pure virtual]

Gets the message priority level.

The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

### Returns

priority value

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 402), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 402).

6.515.3.13 `virtual bool cms::Message::getCMSRedelivered ( ) const throw ( CMSEException )`  
[pure virtual]

Gets an indication of whether this message is being redelivered.

If a client receives a message with the `CMSRedelivered` field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

**Returns**

true if this message is being redelivered

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 402), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 402), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 402).

6.515.3.14 `virtual const cms::Destination* cms::Message::getCMSReplyTo ( ) const throw ( CMSEException )` [pure virtual]

Gets the **Destination** (p. 1688) object to which a reply to this message should be sent.

**Returns**

**Destination** (p. 1688) to which to send a response to this message

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 403), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 403).

6.515.3.15 `virtual long long cms::Message::getCMSTimestamp ( ) const throw ( CMSException ) [pure virtual]`

Gets the message timestamp.

The CMSTimestamp header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, CMSTimestamp is ignored. When the send or publish method returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.

Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the `MessageProducer.setDisableMessageTimestamp` method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

### Returns

the message timestamp

### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 403), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 403).



6.515.3.16 `virtual std::string cms::Message::getCMSType ( ) const throw ( CMSEException )`  
[pure virtual]

Gets the message type identifier supplied by the client when the message was sent.

#### Returns

the message type

#### See also

`setCMSType` (p. 2516)

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 403), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 403), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 403).

6.515.3.17 `virtual double cms::Message::getDoubleProperty ( const std::string & name )`  
`const throw ( MessageFormatException, CMSEException )` [pure virtual]

Gets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`

(p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 404), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 404).

6.515.3.18 `virtual float cms::Message::getFloatProperty ( const std::string & name ) const throw ( MessageFormatException, CMSException ) [pure virtual]`

Gets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	if the property does not exist.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`  
 (p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 404), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 404), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 404).

6.515.3.19 `virtual int cms::Message::getIntProperty ( const std::string & name ) const throw ( MessageFormatException, CMSException ) [pure virtual]`

Gets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	if the property does not exist.
----------------------------------	---------------------------------

<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
---	---------------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 405), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 405).

6.515.3.20 `virtual long long cms::Message::getLongProperty ( const std::string & name ) const throw ( MessageFormatException, CMSEException ) [pure virtual]`

Gets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 405), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 405), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 405).

6.515.3.21 `virtual std::vector<std::string> cms::Message::getPropertyNames ( ) const throw ( CMSEException ) [pure virtual]`

Retrieves the property names.

**Returns**

The complete set of property names currently in this message.

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if an internal error occurs.
---	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 406), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 406).

```
6.515.3.22 virtual short cms::Message::getShortProperty ( const std::string & name )
            const throw ( MessageFormatException, CMSException ) [pure
            virtual]
```

Gets a short property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	if the property does not exist.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 406), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 406).

6.515.3.23 `virtual std::string cms::Message::getStringProperty ( const std::string & name )  
const throw ( MessageFormatException, CMSEException ) [pure  
virtual]`

Gets a string property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

#### Returns

The value for the named property.

#### Exceptions

<b>CMSEException</b> (p. 1130)	if the property does not exist.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 406), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 406), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 406).

6.515.3.24 `virtual bool cms::Message::propertyExists ( const std::string & name ) const throw ( CMSEException ) [pure virtual]`

Indicates whether or not a given property exists.

#### Parameters

<i>name</i>	The name of the property to look up.
-------------	--------------------------------------

#### Returns

True if the property exists in this message.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`

(p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 407), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 407).

6.515.3.25 `virtual void cms::Message::setBooleanProperty ( const std::string & name, bool  
 value ) throw ( MessageNotWriteableException, CMSException ) [pure  
 virtual]`

Sets a boolean property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2680)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::Message >`  
 (p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 407), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 407), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 407).

6.515.3.26 `virtual void cms::Message::setByteProperty ( const std::string & name, unsigned  
 char value ) throw ( MessageNotWriteableException, CMSException )  
 [pure virtual]`

Sets a byte property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2680)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 408), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 408).

**6.515.3.27** `virtual void cms::Message::setCMSCorrelationID ( const std::string & correlationId )`  
`throw ( CMSException ) [pure virtual]`

Sets the correlation ID for the message.

A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMSCorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A byte[] value is used for this purpose. CMS providers without native correlation ID values are not required to support byte[] values. The use of a byte[] value for CMSCorrelationID is non-portable.

#### Parameters

<i>correlationId</i>	The message ID of a message being referred to.
----------------------	--

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs.
----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >

(p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 408), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 408).

6.515.3.28 `virtual void cms::Message::setCMSDeliveryMode ( int mode ) throw ( CMSException ) [pure virtual]`

Sets the **DeliveryMode** (p. 1687) for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

#### Parameters

<i>mode</i>	<b>DeliveryMode</b> (p. 1687) enumerated value.
-------------	---

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs.
----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 408), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 408), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 408).

6.515.3.29 `virtual void cms::Message::setCMSDestination ( const Destination * destination ) throw ( CMSException ) [pure virtual]`

Sets the **Destination** (p. 1688) object for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

#### Parameters

<i>destination</i>	<b>Destination</b> (p. 1688) Object
--------------------	-------------------------------------

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs.
----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >



(p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 409), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 409).

6.515.3.30 `virtual void cms::Message::setCMSExpiration ( long long expireTime ) throw ( CMSException )` [pure virtual]

Sets the message's expiration value.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

#### Parameters

<i>expireTime</i>	the message's expiration time
-------------------	-------------------------------

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs.
----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 409), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 409), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 409).

6.515.3.31 `virtual void cms::Message::setCMSMessageID ( const std::string & id ) throw ( CMSException )` [pure virtual]

Sets the message ID.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

#### Parameters

<i>id</i>	the ID of the message
-----------	-----------------------

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs.
----------------------------------	--------------------------------

6.515.3.32 `virtual void cms::Message::setCMSPriority ( int priority ) throw ( CMSEException )`  
`[pure virtual]`

Sets the Priority Value for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

#### Parameters

<i>priority</i>	priority value for this message
-----------------	---------------------------------

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 410), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 410).

6.515.3.33 `virtual void cms::Message::setCMSRedelivered ( bool redelivered ) throw (`  
`CMSEException ) [pure virtual]`

Specifies whether this message is being redelivered.

This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

#### Parameters

<i>redelivered</i>	boolean redelivered value
--------------------	---------------------------

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage`  
 > (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage`  
 > (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::Message` >  
 (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage`  
 > (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage`  
 > (p. 410), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
 > (p. 410).

6.515.3.34 `virtual void cms::Message::setCMSReplyTo ( const cms::Destination *  
destination ) throw ( CMSEException ) [pure virtual]`

Sets the **Destination** (p. 1688) object to which a reply to this message should be sent.

The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 3093) object or a **Topic** (p. 3757) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

#### Parameters

<i>destination</i>	<b>Destination</b> (p. 1688) to which to send a response to this message
--------------------	--

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 410), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 410), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 410).

6.515.3.35 `virtual void cms::Message::setCMSTimestamp ( long long timeStamp ) throw ( CMSEException ) [pure virtual]`

Sets the message timestamp.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

#### Parameters

<i>timeStamp</i>	integer time stamp value
------------------	--------------------------

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 411), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 411).

**6.515.3.36** `virtual void cms::Message::setCMSType ( const std::string & type ) throw ( CMSEException )` [pure virtual]

Sets the message type.

Some CMS providers use a message repository that contains the definitions of messages sent by applications. The CMSType header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to CMSType, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for CMSType that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

#### Parameters

<i>type</i>	the message type
-------------	------------------

#### See also

`getCMSType` (p. 2505)

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 411), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 411), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 411).

6.515.3.37 virtual void cms::Message::setDoubleProperty ( const std::string & *name*, double *value* ) throw ( MessageNotWriteableException, CMSException ) [pure virtual]

Sets a double property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2680)	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**< cms::BytesMessage > (p. 411), **activemq::commands::ActiveMQMessageTemplate**< cms::MapMessage > (p. 411), **activemq::commands::ActiveMQMessageTemplate**< cms::Message > (p. 411), **activemq::commands::ActiveMQMessageTemplate**< cms::StreamMessage > (p. 411), **activemq::commands::ActiveMQMessageTemplate**< cms::TextMessage > (p. 411), and **activemq::commands::ActiveMQMessageTemplate**< cms::ObjectMessage > (p. 411).

6.515.3.38 virtual void cms::Message::setFloatProperty ( const std::string & *name*, float *value* ) throw ( MessageNotWriteableException, CMSException ) [pure virtual]

Sets a float property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2680)	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**< cms::BytesMessage > (p. 412), **activemq::commands::ActiveMQMessageTemplate**< cms::MapMessage > (p. 412), **activemq::commands::ActiveMQMessageTemplate**< cms::Message > (p. 412), **activemq::commands::ActiveMQMessageTemplate**< cms::StreamMessage

> (p. 412), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage**  
 > (p. 412), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage**  
 > (p. 412).

6.515.3.39 `virtual void cms::Message::setIntProperty ( const std::string & name, int value  
 ) throw ( MessageNotWriteableException, CMSException ) [pure  
 virtual]`

Sets a int property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2680)	- if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage**  
 > (p. 412), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage**  
 > (p. 412), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** >  
 (p. 412), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage**  
 > (p. 412), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage**  
 > (p. 412), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage**  
 > (p. 412).

6.515.3.40 `virtual void cms::Message::setLongProperty ( const std::string & name, long long  
 value ) throw ( MessageNotWriteableException, CMSException ) [pure  
 virtual]`

Sets a long property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2680)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 412), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 412), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 412), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 412), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 412), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 412).

6.515.3.41 `virtual void cms::Message::setShortProperty ( const std::string & name, short value ) throw ( MessageNotWriteableException, CMSException ) [pure virtual]`

Sets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
<b>MessageNotWriteableException</b> (p. 2680)	- if properties are read-only

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 413), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 413).

6.515.3.42 `virtual void cms::Message::setStringProperty ( const std::string & name, const std::string & value ) throw ( MessageNotWriteableException, CMSException ) [pure virtual]`

Sets a string property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the name is an empty string.
----------------------------------	-----------------------------------

<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if properties are read-only
---	-------------------------------

Implemented in `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 413), `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 413), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 413).

The documentation for this class was generated from the following file:

- `src/main/cms/Message.h`

## 6.516 `activemq::commands::MessageAck` Class Reference

```
#include <src/main/activemq/commands/MessageAck.h>
```

Inheritance diagram for `activemq::commands::MessageAck`:

### Public Member Functions

- **MessageAck** ()
- virtual **~MessageAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)



- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char ackType)
- virtual const **Pointer**< **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &firstMessageId)
- virtual const **Pointer**< **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &lastMessageId)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int messageCount)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGEACK** = 22

### Protected Attributes

- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **TransactionId** > transactionId
- **Pointer**< **ConsumerId** > consumerId
- unsigned char ackType
- **Pointer**< **MessageId** > firstMessageId
- **Pointer**< **MessageId** > lastMessageId
- int messageCount

## 6.516.1 Constructor & Destructor Documentation

6.516.1.1 activemq::commands::MessageAck::MessageAck ( )

6.516.1.2 virtual activemq::commands::MessageAck::~~MessageAck ( ) [virtual]

## 6.516.2 Member Function Documentation

6.516.2.1 `virtual MessageAck* activemq::commands::MessageAck::cloneDataStructure ( )`  
`const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.516.2.2 `virtual void activemq::commands::MessageAck::copyDataStructure ( const DataStructure * src )`  
`[virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.516.2.3 `virtual bool activemq::commands::MessageAck::equals ( const DataStructure * value ) const`  
`[virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.516.2.4 `virtual unsigned char activemq::commands::MessageAck::getAckType ( ) const`  
`[virtual]`

6.516.2.5 `virtual const Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ( ) const`  
`[virtual]`

6.516.2.6 `virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId ( )`  
`[virtual]`

6.516.2.7 virtual unsigned char activemq::commands::MessageAck::getDataStructureType ( )  
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.516.2.8 virtual const Pointer<ActiveMQDestination>&  
activemq::commands::MessageAck::getDestination ( ) const [virtual]

6.516.2.9 virtual Pointer<ActiveMQDestination>&  
activemq::commands::MessageAck::getDestination ( ) [virtual]

6.516.2.10 virtual const Pointer<MessageId>& ac-  
tivemq::commands::MessageAck::getFirstMessageId ( ) const  
[virtual]

6.516.2.11 virtual Pointer<MessageId>& ac-  
tivemq::commands::MessageAck::getFirstMessageId ( )  
[virtual]

6.516.2.12 virtual const Pointer<MessageId>& ac-  
tivemq::commands::MessageAck::getLastMessageId ( ) const  
[virtual]

6.516.2.13 virtual Pointer<MessageId>& ac-  
tivemq::commands::MessageAck::getLastMessageId ( )  
[virtual]

6.516.2.14 virtual int activemq::commands::MessageAck::getMessageCount ( ) const  
[virtual]

6.516.2.15 virtual const Pointer<TransactionId>& ac-  
tivemq::commands::MessageAck::getTransactionId ( ) const  
[virtual]

6.516.2.16 virtual Pointer<TransactionId>& ac-  
tivemq::commands::MessageAck::getTransactionId ( )  
[virtual]

6.516.2.17 virtual bool activemq::commands::MessageAck::isMessageAck ( ) const  
[inline, virtual]

#### Returns

an answer of true to the **isMessageAck()** (p. 2524) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 727).

- 6.516.2.18 `virtual void activemq::commands::MessageAck::setAckType ( unsigned char  
ackType ) [virtual]`
- 6.516.2.19 `virtual void activemq::commands::MessageAck::setConsumerId ( const Pointer<  
ConsumerId > & consumerId ) [virtual]`
- 6.516.2.20 `virtual void activemq::commands::MessageAck::setDestination ( const Pointer<  
ActiveMQDestination > & destination ) [virtual]`
- 6.516.2.21 `virtual void activemq::commands::MessageAck::setFirstMessagId ( const  
Pointer< MessageId > & firstMessagId ) [virtual]`
- 6.516.2.22 `virtual void activemq::commands::MessageAck::setLastMessagId ( const  
Pointer< MessageId > & lastMessagId ) [virtual]`
- 6.516.2.23 `virtual void activemq::commands::MessageAck::setMessageCount ( int  
messageCount ) [virtual]`
- 6.516.2.24 `virtual void activemq::commands::MessageAck::setTransactionId ( const Pointer<  
TransactionId > & transactionId ) [virtual]`
- 6.516.2.25 `virtual std::string activemq::commands::MessageAck::toString ( ) const  
[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

- 6.516.2.26 `virtual Pointer<Command> activemq::commands::MessageAck::visit  
( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.516.3 Field Documentation

- 6.516.3.1 unsigned char activemq::commands::MessageAck::ackType  
[protected]
- 6.516.3.2 Pointer<ConsumerId> activemq::commands::MessageAck::consumerId  
[protected]
- 6.516.3.3 Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination  
[protected]
- 6.516.3.4 Pointer<MessageId> activemq::commands::MessageAck::firstMessageId  
[protected]
- 6.516.3.5 const unsigned char activemq::commands::MessageAck::ID\_ - MESSAGEACK = 22 [static]
- 6.516.3.6 Pointer<MessageId> activemq::commands::MessageAck::lastMessageId  
[protected]
- 6.516.3.7 int activemq::commands::MessageAck::messageCount  
[protected]
- 6.516.3.8 Pointer<TransactionId> activemq::commands::MessageAck::transactionId  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageAck.h**

## 6.517 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2526).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller:

### Public Member Functions

- **MessageAckMarshaller** ()
- virtual ~**MessageAckMarshaller** ()

- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.517.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2526).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.517.2 Constructor & Destructor Documentation

- 6.517.2.1 **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::MessageAckMarshaller**  
( ) [*inline*]
- 6.517.2.2 **virtual activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::~~MessageAckMarshaller**  
( ) [*inline, virtual*]

### 6.517.3 Member Function Documentation

- 6.517.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::createObject** ( )  
**const** [*virtual*]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.517.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.517.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.517.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.517.3.5  virtual int activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.517.3.6  virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.518 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller

### Class Reference 2539

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

6.517.3.7 virtual void activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**MessageAckMarshaller.h**

## 6.518 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller

### Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2530).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller:

#### Public Member Functions

- **MessageAckMarshaller** ()

- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.518.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2530).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.518.2 Constructor & Destructor Documentation

6.518.2.1 **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::MessageAckMarshaller** ( ) [inline]

6.518.2.2 **virtual activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::~~MessageAckMarshaller** ( ) [inline, virtual]

### 6.518.3 Member Function Documentation

6.518.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.518.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::getDataStructureType ( )const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.518.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.518.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.518.3.5  virtual int activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.518.3.6  virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.519 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller

### Class Reference 2543

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

6.518.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageAckMarshaller.h**

## 6.519 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller

### Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2534).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller:

#### Public Member Functions

- **MessageAckMarshaller** ()

- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.519.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2534).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.519.2 Constructor & Destructor Documentation

- 6.519.2.1 **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::MessageAckMarshaller**  
 ( ) [inline]
- 6.519.2.2 **virtual activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::~~MessageAckMarshaller**  
 ( ) [inline, virtual]

### 6.519.3 Member Function Documentation

- 6.519.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

**6.519.3.2** virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::getDataStructureType ( )const [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.519.3.3** virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

**6.519.3.4** virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.519.3.5  virtual int activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.519.3.6  virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## 6.520 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller

### Class Reference 2547

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

6.519.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageAckMarshaller.h**

## 6.520 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller

### Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2538).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller:

#### Public Member Functions

- **MessageAckMarshaller** ()

- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.520.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2538).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.520.2 Constructor & Destructor Documentation

- 6.520.2.1 **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::MessageAckMarshaller**  
 ( ) [inline]
- 6.520.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::~~MessageAckMarshaller**  
 ( ) [inline, virtual]

### 6.520.3 Member Function Documentation

- 6.520.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.520.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.520.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.520.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.520.3.5  virtual int activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.520.3.6  virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## 6.521 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller

### Class Reference 2551

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

6.520.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**MessageAckMarshaller.h**

## 6.521 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller

### Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2542).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller:

#### Public Member Functions

- **MessageAckMarshaller** ()

- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.521.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2542).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.521.2 Constructor & Destructor Documentation

- 6.521.2.1 **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::MessageAckMarshaller**  
 ( ) [inline]
- 6.521.2.2 **virtual activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::~~MessageAckMarshaller**  
 ( ) [inline, virtual]

### 6.521.3 Member Function Documentation

- 6.521.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.521.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::getDataStructureType  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.521.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseMarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \*  
 dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.521.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseUnmarshal  
 ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure,  
 decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.521.3.5  virtual int activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.521.3.6  virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

6.521.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageAckMarshaller.h**

## 6.522 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2546).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller:

## Public Member Functions

- **MessageAckMarshaller** ()

- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.522.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2546).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.522.2 Constructor & Destructor Documentation

6.522.2.1 **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::MessageAckMarshaller** ( ) [inline]

6.522.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::~~MessageAckMarshaller** ( ) [inline, virtual]

### 6.522.3 Member Function Documentation

6.522.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.522.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

## Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.522.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.522.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.522.3.5  virtual int activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.522.3.6  virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

6.522.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MessageAckMarshaller.h**

**6.523 cms::MessageConsumer Class Reference**

A client uses a **MessageConsumer** (p. 2550) to received messages from a destination.

```
#include <src/main/cms/MessageConsumer.h>
```

Inheritance diagram for cms::MessageConsumer:

**Public Member Functions**

- virtual **~MessageConsumer** ()
- virtual **Message** \* **receive** ()=0 throw ( **CMSEException** )

*Synchronously Receive a **Message** (p. 2493).*

- virtual **Message \* receive** (int millisecs)=0 throw ( CMSEException )

*Synchronously Receive a **Message** (p. 2493), time out after defined interval.*

- virtual **Message \* receiveNoWait** ()=0 throw ( CMSEException )

*Receive a **Message** (p. 2493), does not wait if there isn't a new message to read, returns NULL if nothing read.*

- virtual void **setMessageListener** (MessageListener \*listener)=0 throw ( CMSEException )

*Sets the **MessageListener** (p. 2652) that this class will send notifis on.*

- virtual MessageListener \* **getMessageListener** () const =0 throw ( CMSEException )

*Gets the **MessageListener** (p. 2652) that this class will send mew **Message** (p. 2493) notification events to.*

- virtual std::string **getMessageSelector** () const =0 throw ( cms::CMSEException )

*Gets this message consumer's message selector expression.*

### 6.523.1 Detailed Description

A client uses a **MessageConsumer** (p. 2550) to received messages from a destination.

A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its `receive` methods. There are several variations of `receive` that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a **MessageListener** (p. 2652) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the **MessageListener** (p. 2652)'s `onMessage` method.

When the `MessageConsumer`'s `close` method is called the method can block while an asynchronous message delivery is in progress or until a `receive` operation completes. A blocked consumer in a `receive` call will return a Null when the `close` method is called.

#### See also

**MessageListener** (p. 2652)

#### Since

1.0

### 6.523.2 Constructor & Destructor Documentation

- 6.523.2.1 virtual cms::MessageConsumer::~MessageConsumer ( ) [inline, virtual]

### 6.523.3 Member Function Documentation

6.523.3.1 virtual **MessageListener\*** cms::MessageConsumer::getMessageListener ( ) const  
throw ( **CMSEException** ) [pure virtual]

Gets the **MessageListener** (p. 2652) that this class will send new **Message** (p. 2493) notification events to.

#### Returns

The listener of messages received by this consumer

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1042), and **activemq::core::ActiveMQConsumer** (p. 288).

6.523.3.2 virtual std::string cms::MessageConsumer::getMessageSelector ( ) const throw ( cms::CMSEException ) [pure virtual]

Gets this message consumer's message selector expression.

#### Returns

This Consumer's selector expression or "".

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1042), and **activemq::core::ActiveMQConsumer** (p. 288).

6.523.3.3 virtual **Message\*** cms::MessageConsumer::receive ( int *millisecs* ) throw ( **CMSEException** ) [pure virtual]

Synchronously Receive a **Message** (p. 2493), time out after defined interval.

Returns null if nothing read.

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1043), and **activemq::core::ActiveMQConsumer** (p. 289).

6.523.3.4 **virtual Message\* cms::MessageConsumer::receive ( ) throw ( CMSEException )**  
[pure virtual]

Synchronously Receive a **Message** (p. 2493).

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1043), and **activemq::core::ActiveMQConsumer** (p. 289).

6.523.3.5 **virtual Message\* cms::MessageConsumer::receiveNoWait ( ) throw ( CMSEException )** [pure virtual]

Receive a **Message** (p. 2493), does not wait if there isn't a new message to read, returns NULL if nothing read.

#### Returns

new message which the caller owns and must delete.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1043), and **activemq::core::ActiveMQConsumer** (p. 290).

6.523.3.6 **virtual void cms::MessageConsumer::setMessageListener ( MessageListener \* listener ) throw ( CMSEException )** [pure virtual]

Sets the **MessageListener** (p. 2652) that this class will send notifs on.



**Parameters**

<i>listener</i>	The listener of messages received by this consumer.
-----------------	---

**Exceptions**

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedConsumer** (p. 1044), and **activemq::core::ActiveMQConsumer** (p. 291).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageConsumer.h**

**6.524 activemq::cmsutil::MessageCreator Class Reference**

Creates the user-defined message to be sent by the **CmsTemplate** (p. 1140).

```
#include <src/main/activemq/cmsutil/MessageCreator.h>
```

**Public Member Functions**

- virtual **~MessageCreator** ()
- virtual **cms::Message \* createMessage** (**cms::Session \*session**)=0 throw (**cms::CMSEException** )

*Creates a message from the given session.*

**6.524.1 Detailed Description**

Creates the user-defined message to be sent by the **CmsTemplate** (p. 1140).

**6.524.2 Constructor & Destructor Documentation**

6.524.2.1 virtual **activemq::cmsutil::MessageCreator::~MessageCreator** ( ) [inline, virtual]

**6.524.3 Member Function Documentation**

6.524.3.1 virtual **cms::Message\* activemq::cmsutil::MessageCreator::createMessage** ( **cms::Session \* session** ) throw ( **cms::CMSEException** ) [pure virtual]

Creates a message from the given session.

**Parameters**

<i>session</i>	the CMS <i>Session</i>
----------------	------------------------

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	if thrown by CMS API methods
--	------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**MessageCreator.h**

**6.525 activemq::commands::MessageDispatch Class Reference**

```
#include <src/main/activemq/commands/MessageDispatch.h>
```

Inheritance diagram for `activemq::commands::MessageDispatch`:

**Public Member Functions**

- **MessageDispatch** ()
- virtual **~MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageDispatch \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)

- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &message)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGEDISPATCH** = 21

### Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **Message** > **message**
- int **redeliveryCounter**

## 6.525.1 Constructor & Destructor Documentation

6.525.1.1 **activemq::commands::MessageDispatch::MessageDispatch** ( )

6.525.1.2 **virtual activemq::commands::MessageDispatch::~~MessageDispatch** ( )  
[virtual]

## 6.525.2 Member Function Documentation

6.525.2.1 **virtual MessageDispatch\*** **activemq::commands::MessageDispatch::cloneDataStructure**  
( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.525.2.2 `virtual void activemq::commands::MessageDispatch::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 724).

6.525.2.3 `virtual bool activemq::commands::MessageDispatch::equals ( const DataStructure * value ) const [virtual]`

Compares the `DataStructure` (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 725).

6.525.2.4 `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ( ) const [virtual]`

6.525.2.5 `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ( ) [virtual]`

6.525.2.6 `virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new `DataStructure` (p. 1628) type copy.

Implements `activemq::commands::DataStructure` (p. 1631).

6.525.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ( ) const [virtual]`

- 6.525.2.8 virtual **Pointer**<**ActiveMQDestination**>&  
activemq::commands::MessageDispatch::getDestination ( ) [virtual]
- 6.525.2.9 virtual const **Pointer**<**Message**>& ac-  
tivemq::commands::MessageDispatch::getMessage ( ) const  
[virtual]
- 6.525.2.10 virtual **Pointer**<**Message**>& ac-  
tivemq::commands::MessageDispatch::getMessage ( )  
[virtual]
- 6.525.2.11 virtual int activemq::commands::MessageDispatch::getRedeliveryCounter ( ) const  
[virtual]
- 6.525.2.12 virtual bool activemq::commands::MessageDispatch::isMessageDispatch ( ) const  
[inline, virtual]

### Returns

an answer of true to the **isMessageDispatch()** (p. 2558) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 727).

- 6.525.2.13 virtual void activemq::commands::MessageDispatch::setConsumerId ( const  
**Pointer**< **ConsumerId** > & *consumerId* ) [virtual]
- 6.525.2.14 virtual void activemq::commands::MessageDispatch::setDestination ( const  
**Pointer**< **ActiveMQDestination** > & *destination* ) [virtual]
- 6.525.2.15 virtual void activemq::commands::MessageDispatch::setMessage ( const **Pointer**<  
**Message** > & *message* ) [virtual]
- 6.525.2.16 virtual void activemq::commands::MessageDispatch::setRedeliveryCounter ( int  
*redeliveryCounter* ) [virtual]
- 6.525.2.17 virtual std::string activemq::commands::MessageDispatch::toString ( ) const  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

```
6.525.2.18 virtual Pointer<Command> activemq::commands::MessageDispatch::visit
( activemq::state::CommandVisitor * visitor ) throw (
exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.525.3 Field Documentation

```
6.525.3.1 Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId
[protected]
```

```
6.525.3.2 Pointer<ActiveMQDestination> ac-
tivemq::commands::MessageDispatch::destination
[protected]
```

```
6.525.3.3 const unsigned char activemq::commands::MessageDispatch::ID_ -
MESSAGEDISPATCH = 21 [static]
```

```
6.525.3.4 Pointer<Message> activemq::commands::MessageDispatch::message
[protected]
```

```
6.525.3.5 int activemq::commands::MessageDispatch::redeliveryCounter
[protected]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatch.h**

### 6.526 activemq::core::MessageDispatchChannel Class Reference

```
#include <src/main/activemq/core/MessageDispatchChannel.h>
```

Inheritance diagram for activemq::core::MessageDispatchChannel:

### Public Member Functions

- **MessageDispatchChannel** ()

- virtual **~MessageDispatchChannel** ()
- void **enqueue** (const **Pointer**< **MessageDispatch** > &message)  
*Add a Message to the Channel behind all pending message.*
- void **enqueueFirst** (const **Pointer**< **MessageDispatch** > &message)  
*Add a message to the front of the Channel.*
- bool **isEmpty** () const
- bool **isClosed** () const
- bool **isRunning** () const
- **Pointer**< **MessageDispatch** > **dequeue** (long long timeout) throw ( exceptions::ActiveMQException )  
*Used to get an enqueued message.*
- **Pointer**< **MessageDispatch** > **dequeueNoWait** ()  
*Used to get an enqueued message if there is one queued right now.*
- **Pointer**< **MessageDispatch** > **peek** () const  
*Peek in the Queue and return the first message in the Channel without removing it from the channel.*
- void **start** ()  
*Starts dispatch of messages from the Channel.*
- void **stop** ()  
*Stops dispatch of message from the Channel.*
- void **close** ()  
*Close this channel no messages will be dispatched after this method is called.*
- void **clear** ()  
*Clear the Channel, all pending messages are removed.*
- int **size** () const
- std::vector< **Pointer**< **MessageDispatch** > > **removeAll** ()  
*Remove all messages that are currently in the Channel and return them as a list of Messages.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.526.1 Constructor & Destructor Documentation

6.526.1.1 **activemq::core::MessageDispatchChannel::MessageDispatchChannel ( )**

6.526.1.2 **virtual activemq::core::MessageDispatchChannel::~~MessageDispatchChannel ( )**  
[virtual]

### 6.526.2 Member Function Documentation

6.526.2.1 **void activemq::core::MessageDispatchChannel::clear ( )**

Clear the Channel, all pending messages are removed.

6.526.2.2 **void activemq::core::MessageDispatchChannel::close ( )**

Close this channel no messages will be dispatched after this method is called.

6.526.2.3 **Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue ( long long timeout ) throw ( exceptions::ActiveMQException )**

Used to get an enqueued message.

The amount of time this method blocks is based on the timeout value. - if timeout==-1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

#### Returns

null if we timeout or if the consumer is closed.

#### Exceptions

<i>ActiveMQException</i>
--------------------------



6.526.2.4 **Pointer<MessageDispatch>** **activemq::core::MessageDispatchChannel::dequeueNoWait (**  
    **)**

Used to get an enqueued message if there is one queued right now.

If there is no waiting message than this method returns Null.

#### Returns

a message if there is one in the queue.

6.526.2.5 **void** **activemq::core::MessageDispatchChannel::enqueue ( const Pointer<**  
    **MessageDispatch > & message )**

Add a Message to the Channel behind all pending message.

#### Parameters

<i>message</i>	- The message to add to the Channel.
----------------	--------------------------------------

6.526.2.6 **void** **activemq::core::MessageDispatchChannel::enqueueFirst ( const Pointer<**  
    **MessageDispatch > & message )**

Add a message to the front of the Channel.

#### Parameters

<i>message</i>	- The Message to add to the front of the Channel.
----------------	---

6.526.2.7 **bool** **activemq::core::MessageDispatchChannel::isClosed ( ) const** [inline]

#### Returns

has the Queue been closed.

6.526.2.8 **bool** **activemq::core::MessageDispatchChannel::isEmpty ( ) const**

#### Returns

true if there are no messages in the Channel.

6.526.2.9 **bool** **activemq::core::MessageDispatchChannel::isRunning ( ) const** [inline]

#### Returns

true if the Channel currently running and will dequeue message.

6.526.2.10 `virtual void activemq::core::MessageDispatchChannel::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

6.526.2.11 `virtual void activemq::core::MessageDispatchChannel::notify ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

6.526.2.12 `virtual void activemq::core::MessageDispatchChannel::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

6.526.2.13 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek ( ) const`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

#### Returns

a message if there is one in the queue.

6.526.2.14 `std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ( )`

Remove all messages that are currently in the Channel and return them as a list of Messages.

#### Returns

a list of Messages that was previously in the Channel.

6.526.2.15 `int activemq::core::MessageDispatchChannel::size ( ) const`

#### Returns

the number of Messages currently in the Channel.

6.526.2.16 `void activemq::core::MessageDispatchChannel::start ( )`

Starts dispatch of messages from the Channel.

6.526.2.17 `void activemq::core::MessageDispatchChannel::stop ( )`

Stops dispatch of message from the Channel.

6.526.2.18 `virtual bool activemq::core::MessageDispatchChannel::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

6.526.2.19 `virtual void activemq::core::MessageDispatchChannel::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

6.526.2.20 `virtual void activemq::core::MessageDispatchChannel::wait ( long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

6.526.2.21 `virtual void activemq::core::MessageDispatchChannel::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

```
6.526.2.22 virtual void activemq::core::MessageDispatchChannel::wait ( )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException,
                    decaf::lang::exceptions::InterruptedException ) [inline,
                                virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**MessageDispatchChannel.h**

## 6.527 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller

### Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2566).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller:

#### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.527.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2566).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.527.2 Constructor & Destructor Documentation

6.527.2.1 `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]`

6.527.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::~~MessageDispatchMarshaller ( ) [inline, virtual]`

## 6.527.3 Member Function Documentation

6.527.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.527.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.527.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.527.3.4  virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.527.3.5  virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.527 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller

### Class Reference 2579

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.527.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.527.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchMarshaller.h**

## 6.528 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2570).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller:

### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.528.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2570).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.528.2 Constructor & Destructor Documentation

6.528.2.1 `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]`

6.528.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::~~MessageDispatchMarshaller ( ) [inline, virtual]`

## 6.528.3 Member Function Documentation

6.528.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.528.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.528.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.528.3.4  virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.528.3.5  virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.528 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller

### Class Reference 2583

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.528.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.528.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageDispatchMarshaller.h**

## 6.529 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2574).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller:

### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.529.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2574).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.529.2 Constructor & Destructor Documentation

6.529.2.1 `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]`

6.529.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::~~MessageDispatchMarshaller ( ) [inline, virtual]`

## 6.529.3 Member Function Documentation

6.529.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.529.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.529.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.529.3.4  virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.529.3.5  virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.529 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller

### Class Reference 2587

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.529.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.529.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MessageDispatchMarshaller.h**

## 6.530 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2578).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller:

### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.530.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2578).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.530.2 Constructor & Destructor Documentation

6.530.2.1 **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::MessageDispatchMarshaller**  
( ) [inline]

6.530.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::~~MessageDispatchMarshaller**  
( ) [inline, virtual]

## 6.530.3 Member Function Documentation

6.530.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.530.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.530.3.3 **virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseMarshal**  
( **OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataOutputStream \* dataOut** ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.530.3.4  virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.530.3.5  virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.530 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller

### Class Reference 2591

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.530.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.530.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**MessageDispatchMarshaller.h**

## 6.531 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2582).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller:

### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.531.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2582).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.531.2 Constructor & Destructor Documentation

6.531.2.1 `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]`

6.531.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::~~MessageDispatchMarshaller ( ) [inline, virtual]`

## 6.531.3 Member Function Documentation

6.531.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.531.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.531.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.531.3.4  virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.531.3.5  virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.531 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller

### Class Reference 2595

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.531.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.531.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchMarshaller.h**

## 6.532 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2586).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller:

### Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.532.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2586).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.532.2 Constructor & Destructor Documentation

6.532.2.1 `activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::MessageDispatchMarshaller ( ) [inline]`

6.532.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::~~MessageDispatchMarshaller ( ) [inline, virtual]`

## 6.532.3 Member Function Documentation

6.532.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.532.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.532.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

```
6.532.3.4  virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.532.3.5  virtual int activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.532 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller

### Class Reference 2599

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.532.3.6 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.532.3.7 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**MessageDispatchMarshaller.h**

## 6.533 activemq::commands::MessageDispatchNotification Class Reference

```
#include <src/main/activemq/commands/MessageDispatchNotification.h>
```

Inheritance diagram for activemq::commands::MessageDispatchNotification:

### Public Member Functions

- **MessageDispatchNotification** ()
- virtual **~MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageDispatchNotification \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long deliverySequenceId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## 6.533 activemq::commands::MessageDispatchNotification Class Reference 2601

### Static Public Attributes

- static const unsigned char **ID\_MESSAGE\_DISPATCH\_NOTIFICATION** = 90

### Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **deliverySequenceId**
- **Pointer**< **MessageId** > **messageId**

### 6.533.1 Constructor & Destructor Documentation

6.533.1.1 `activemq::commands::MessageDispatchNotification::MessageDispatchNotification ( )`

6.533.1.2 `virtual activemq::commands::MessageDispatchNotification::~~MessageDispatchNotification ( ) [virtual]`

### 6.533.2 Member Function Documentation

6.533.2.1 `virtual MessageDispatchNotification* activemq::commands::MessageDispatchNotification::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.533.2.2 `virtual void activemq::commands::MessageDispatchNotification::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.533.2.3 `virtual bool activemq::commands::MessageDispatchNotification::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.533.2.4 `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ( ) const` [virtual]

6.533.2.5 `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId ( )` [virtual]

6.533.2.6 `virtual unsigned char activemq::commands::MessageDispatchNotification::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.533.2.7 `virtual long long activemq::commands::MessageDispatchNotification::getDeliverySequenceId ( ) const` [virtual]

6.533.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ( ) const` [virtual]

6.533.2.9 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination ( )` [virtual]

6.533.2.10 `virtual Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId ( )` [virtual]



## 6.533 activemq::commands::MessageDispatchNotification Class Reference 2603

6.533.2.11 virtual const **Pointer**<**MessageId**>& **activemq::commands::MessageDispatchNotification::getMessageId** ( ) const  
[virtual]

6.533.2.12 virtual bool **activemq::commands::MessageDispatchNotification::isMessageDispatchNotification**  
( ) const [inline, virtual]

### Returns

an answer of true to the **isMessageDispatchNotification()** (p. 2593) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 727).

6.533.2.13 virtual void **activemq::commands::MessageDispatchNotification::setConsumerId** (  
const **Pointer**< **ConsumerId** > & *consumerId* ) [virtual]

6.533.2.14 virtual void **activemq::commands::MessageDispatchNotification::setDeliverySequenceId**  
( long long *deliverySequenceId* ) [virtual]

6.533.2.15 virtual void **activemq::commands::MessageDispatchNotification::setDestination** (  
const **Pointer**< **ActiveMQDestination** > & *destination* ) [virtual]

6.533.2.16 virtual void **activemq::commands::MessageDispatchNotification::setMessageId** (  
const **Pointer**< **MessageId** > & *messageId* ) [virtual]

6.533.2.17 virtual std::string **activemq::commands::MessageDispatchNotification::toString** ( )  
const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.533.2.18 virtual **Pointer**<**Command**> **activemq::commands::MessageDispatchNotification::visit** (  
**activemq::state::CommandVisitor** \* *visitor* ) throw (  
**exceptions::ActiveMQException** ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.533.3 Field Documentation

- 6.533.3.1 **Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId**  
[protected]
- 6.533.3.2 **long long activemq::commands::MessageDispatchNotification::deliverySequenceId**  
[protected]
- 6.533.3.3 **Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination**  
[protected]
- 6.533.3.4 **const unsigned char activemq::commands::MessageDispatchNotification::ID\_ -  
MESSAGEDISPATCHNOTIFICATION = 90** [static]
- 6.533.3.5 **Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatchNotification.h**

## 6.534 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2595).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**:

### Public Member Functions

- **MessageDispatchNotificationMarshaller ()**
- virtual **~MessageDispatchNotificationMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaller.*

## 6.534 ac-

**activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

### Class Reference

2605

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.534.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2595).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.534.2 Constructor & Destructor Documentation

6.534.2.1 **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller** ( ) [inline]

6.534.2.2 **virtual activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller** ( ) [inline, virtual]

### 6.534.3 Member Function Documentation

6.534.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.534.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.534.3.3  virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

```
6.534.3.4  virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**6.534 ac-**  
**ativemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**  
**Class Reference** **2607**  
**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 759).

**6.534.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**  
(p. 760).

**6.534.3.6** `virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.534.3.7 virtual void activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**MessageDispatchNotificationMarshaller.h**

## 6.535 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2599).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**:

#### Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
  - virtual **~MessageDispatchNotificationMarshaller** ()
  - virtual **commands::DataStructure \* createObject** () const
- Creates a new instance of this marshalable type.*

## 6.535 ac-

**activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**

### Class Reference

2609

- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.535.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2599).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.535.2 Constructor & Destructor Documentation

6.535.2.1 **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller**  
( ) [inline]

6.535.2.2 **virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller**  
( ) [inline, virtual]

### 6.535.3 Member Function Documentation

6.535.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.535.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::getDataStructureType() const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.535.3.3  virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.535.3.4  virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.535 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller

### Class Reference 2611

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.535.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.535.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 768).

6.535.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 769).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h`

6.536

activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller

Class Reference

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2603).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h>`

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`:

Public Member Functions

- `MessageDispatchNotificationMarshaller ()`

## 6.536 ac-

activemq:wireformat:openwire::marshal:v3::MessageDispatchNotificationMarshaller

### Class Reference

2613

- virtual `~MessageDispatchNotificationMarshaller ()`
- virtual `commands::DataStructure * createObject ()` const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType ()` const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)` throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)` throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)` throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)` throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)` throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.536.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2603).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.536.2 Constructor & Destructor Documentation

6.536.2.1 `activemq:wireformat:openwire::marshal:v3::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller ( )` [inline]

6.536.2.2 `virtual activemq:wireformat:openwire::marshal:v3::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller ( )` [inline, virtual]

### 6.536.3 Member Function Documentation

```
6.536.3.1 virtual commands::DataStructure* ac-
          tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::createObject
          ( ) const [virtual]
```

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.536.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::getDat
          aStructureType ( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.536.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseMarshal
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
            decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.536 ac-

activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller

## Class Reference

2615

```
6.536.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.536.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

6.536.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 734).

6.536.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 736).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h`

6.537 ac-

activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller

Class Reference

6.537 — activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller 2617

## Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2607).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller:

### Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal**1 (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal**2 (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.537.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2607).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.537.2 Constructor & Destructor Documentation

6.537.2.1 `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller ( ) [inline]`

6.537.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller ( ) [inline, virtual]`

### 6.537.3 Member Function Documentation

6.537.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.537.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.537.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



**6.537 ac-**  
**activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**  
**Class Reference** **2619**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

**6.537.3.4** `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

**6.537.3.5** `virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.537.3.6  virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.537.3.7  virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**MessageDispatchNotificationMarshaller.h**

6.538 ac-

activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller

Class Reference

6.538 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller

## Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2611).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller:

## Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.538.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2611).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.538.2 Constructor & Destructor Documentation

6.538.2.1 `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller ( ) [inline]`

6.538.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller ( ) [inline, virtual]`

### 6.538.3 Member Function Documentation

6.538.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.538.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.538.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**6.538 ac-**  
**activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**  
**Class Reference** **2623**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
 (p. 745).

**6.538.3.4** `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
 (p. 746).

**6.538.3.5** `virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.538.3.6  virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.538.3.7  virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchNotificationMarshaller.h**

6.539 ac-

activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller

Class Reference

6.539 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller

## Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2616).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller:

### Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal**1 (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal**2 (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

#### 6.539.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2616).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.539.2 Constructor & Destructor Documentation

6.539.2.1 `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller ( ) [inline]`

6.539.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::~~MessageDispatchNotificationMarshaller ( ) [inline, virtual]`

## 6.539.3 Member Function Documentation

6.539.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.539.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.539.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



**6.539 ac-**  
**activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**  
**Class Reference** **2627**  
**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

**6.539.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

**6.539.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.539.3.6  virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.539.3.7  virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MessageDispatchNotificationMarshaller.h**

## 6.540 cms::MessageEnumeration Class Reference

Defines an object that enumerates a collection of Messages.

```
#include <src/main/cms/MessageEnumeration.h>
```

Inheritance diagram for cms::MessageEnumeration:

### Public Member Functions

- virtual **~MessageEnumeration** ()
- virtual bool **hasMoreMessages** ()=0  
*Returns true if there are more **Message** (p. 2493) in the Browser that can be retrieved via the `nextMessage` method.*
- virtual **cms::Message \* nextMessage** ()=0 throw ( cms::CMSEException )  
*Returns the Next **Message** (p. 2493) in the **Queue** (p. 3093) if one is present, if no more Message's are available then an Exception is thrown.*

### 6.540.1 Detailed Description

Defines an object that enumerates a collection of Messages.

The client calls the `hasMoreMessages` method to determine if a **Message** (p. 2493) is available. If a **Message** (p. 2493) is available the client calls the `nextMessage` method to retrieve that **Message** (p. 2493), calling `nextMessage` when a **Message** (p. 2493) is not available results in an exception.

Since

2.1

### 6.540.2 Constructor & Destructor Documentation

6.540.2.1 `virtual cms::MessageEnumeration::~~MessageEnumeration ( ) [inline, virtual]`

### 6.540.3 Member Function Documentation

6.540.3.1 `virtual bool cms::MessageEnumeration::hasMoreMessages ( ) [pure virtual]`

Returns true if there are more **Message** (p. 2493) in the Browser that can be retrieved via the `nextMessage` method.

If this method returns false and the `nextMessage` method is called then an Exception will be thrown.

**Returns**

true if more Message's are available in the Browser.

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 459).

6.540.3.2 **virtual cms::Message\* cms::MessageEnumeration::nextMessage ( ) throw ( cms::CMSException )** [pure virtual]

Returns the Next **Message** (p. 2493) in the **Queue** (p. 3093) if one is present, if no more Message's are available then an Exception is thrown.

If a **Message** (p. 2493) object pointer is returned then that object becomes the property of the caller and must be deleted by the caller when finished.

**Returns**

The next **Message** (p. 2493) in the **Queue** (p. 3093).

**Exceptions**

<b>CMSException</b> (p. 1130)	if no more Message's currently in the <b>Queue</b> (p. 3093).
----------------------------------	---

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 460).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageEnumeration.h**

**6.541 cms::MessageEOFException Class Reference**

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3595) or **BytesMessage** (p. 1023) is being read.

```
#include <src/main/cms/MessageEOFException.h>
```

Inheritance diagram for cms::MessageEOFException:

**Public Member Functions**

- **MessageEOFException** () throw ()
- **MessageEOFException** (const **MessageEOFException** &ex) throw ()
- **MessageEOFException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageEOFException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageEOFException** () throw ()

### 6.541.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3595) or **BytesMessage** (p. 1023) is being read.

#### Since

1.3

### 6.541.2 Constructor & Destructor Documentation

6.541.2.1 cms::MessageEOFException::MessageEOFException ( ) throw ()

6.541.2.2 cms::MessageEOFException::MessageEOFException ( const MessageEOFException & ex ) throw ()

6.541.2.3 cms::MessageEOFException::MessageEOFException ( const std::string & message, const std::exception \* cause ) throw ()

6.541.2.4 cms::MessageEOFException::MessageEOFException ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.541.2.5 virtual cms::MessageEOFException::~MessageEOFException ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageEOFException.h**

## 6.542 cms::MessageFormatException Class Reference

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

```
#include <src/main/cms/MessageFormatException.h>
```

Inheritance diagram for cms::MessageFormatException:

### Public Member Functions

- **MessageFormatException** ( ) throw ()
- **MessageFormatException** (const **MessageFormatException** &ex) throw ()
- **MessageFormatException** (const std::string &message, const std::exception \*cause) throw ()

- **MessageFormatException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageFormatException** () throw ()

### 6.542.1 Detailed Description

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if **StreamMessage.readShort** (p. 3603) is used to read a boolean value.

#### Since

1.3

### 6.542.2 Constructor & Destructor Documentation

6.542.2.1 cms::MessageFormatException::MessageFormatException ( ) throw ()

6.542.2.2 cms::MessageFormatException::MessageFormatException ( const MessageFormatException & ex ) throw ()

6.542.2.3 cms::MessageFormatException::MessageFormatException ( const std::string & message, const std::exception \* cause ) throw ()

6.542.2.4 cms::MessageFormatException::MessageFormatException ( const std::string & message, const std::exception \* cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()

6.542.2.5 virtual cms::MessageFormatException::~~MessageFormatException ( ) throw ()  
[virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageFormatException.h**

## 6.543 activemq::commands::MessageId Class Reference

```
#include <src/main/activemq/commands/MessageId.h>
```

Inheritance diagram for activemq::commands::MessageId:

## Public Types

- typedef **decaf::lang::PointerComparator**< **MessageId** > **COMPARATOR**

## Public Member Functions

- **MessageId** ()
- **MessageId** (const **MessageId** &other)
- **MessageId** (const std::string &messageKey)
- **MessageId** (const **Pointer**< **ProducerInfo** > &producerInfo, long long **producerSequenceld**)
- **MessageId** (const **Pointer**< **ProducerId** > &producerId, long long **producerSequenceld**)
- **MessageId** (const std::string &producerId, long long **producerSequenceld**)
- virtual ~**MessageId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessageId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- void **setValue** (const std::string &key)
- void **setTextView** (const std::string &key)
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual long long **getProducerSequenceld** () const
- virtual void **setProducerSequenceld** (long long **producerSequenceld**)
- virtual long long **getBrokerSequenceld** () const
- virtual void **setBrokerSequenceld** (long long **brokerSequenceld**)
- virtual int **compareTo** (const **MessageId** &value) const
- virtual bool **equals** (const **MessageId** &value) const
- virtual bool **operator==** (const **MessageId** &value) const
- virtual bool **operator<** (const **MessageId** &value) const
- **MessageId** & **operator=** (const **MessageId** &other)

## Static Public Attributes

- static const unsigned char **ID\_MESSAGEID** = 110

## Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- long long **producerSequenceId**
- long long **brokerSequenceId**

### 6.543.1 Member Typedef Documentation

- 6.543.1.1 `typedef decaf::lang::PointerComparator<MessageId>  
activemq::commands::MessageId::COMPARATOR`

### 6.543.2 Constructor & Destructor Documentation

- 6.543.2.1 `activemq::commands::MessageId::MessageId ( )`
- 6.543.2.2 `activemq::commands::MessageId::MessageId ( const MessageId & other )`
- 6.543.2.3 `activemq::commands::MessageId::MessageId ( const std::string & messageKey )`
- 6.543.2.4 `activemq::commands::MessageId::MessageId ( const Pointer< ProducerInfo > &  
producerInfo, long long producerSequenceId )`
- 6.543.2.5 `activemq::commands::MessageId::MessageId ( const Pointer< ProducerId > &  
producerId, long long producerSequenceId )`
- 6.543.2.6 `activemq::commands::MessageId::MessageId ( const std::string & producerId, long  
long producerSequenceId )`
- 6.543.2.7 `virtual activemq::commands::MessageId::~~MessageId ( ) [virtual]`

### 6.543.3 Member Function Documentation

- 6.543.3.1 `virtual MessageId* activemq::commands::MessageId::cloneDataStructure ( ) const  
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

## Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1628).



6.543.3.2 `virtual int activemq::commands::MessageId::compareTo ( const MessageId & value ) const [virtual]`

6.543.3.3 `virtual void activemq::commands::MessageId::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.543.3.4 `virtual bool activemq::commands::MessageId::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.543.3.5 `virtual bool activemq::commands::MessageId::equals ( const MessageId & value ) const [virtual]`

6.543.3.6 `virtual long long activemq::commands::MessageId::getBrokerSequenceId ( ) const [virtual]`

6.543.3.7 `virtual unsigned char activemq::commands::MessageId::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

- 6.543.3.8 `virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ( ) const`  
[virtual]
- 6.543.3.9 `virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ( )`  
[virtual]
- 6.543.3.10 `virtual long long activemq::commands::MessageId::getProducerSequenceId ( ) const`  
[virtual]
- 6.543.3.11 `virtual bool activemq::commands::MessageId::operator< ( const MessageId & value ) const`  
[virtual]
- 6.543.3.12 `MessageId& activemq::commands::MessageId::operator= ( const MessageId & other )`
- 6.543.3.13 `virtual bool activemq::commands::MessageId::operator== ( const MessageId & value ) const`  
[virtual]
- 6.543.3.14 `virtual void activemq::commands::MessageId::setBrokerSequenceId ( long long brokerSequenceId )`  
[virtual]
- 6.543.3.15 `virtual void activemq::commands::MessageId::setProducerId ( const Pointer< ProducerId > & producerId )`  
[virtual]
- 6.543.3.16 `virtual void activemq::commands::MessageId::setProducerSequenceId ( long long producerSequenceId )`  
[virtual]
- 6.543.3.17 `void activemq::commands::MessageId::setTextView ( const std::string & key )`
- 6.543.3.18 `void activemq::commands::MessageId::setValue ( const std::string & key )`
- 6.543.3.19 `virtual std::string activemq::commands::MessageId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

### 6.543.4 Field Documentation

- 6.543.4.1 `long long activemq::commands::MessageId::brokerSequenceId`  
[protected]

## 6.544 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference 2637

6.543.4.2 `const unsigned char activemq::commands::MessageId::ID_MESSAGEID = 110 [static]`

6.543.4.3 `Pointer<ProducerId> activemq::commands::MessageId::producerId [protected]`

6.543.4.4 `long long activemq::commands::MessageId::producerSequenceId [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageId.h`

## 6.544 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2628).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`:

### Public Member Functions

- **MessageIdMarshaller** ()
- virtual `~MessageIdMarshaller` ()
- virtual `commands::DataStructure * createObject` () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType` () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.544.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2628).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.544.2 Constructor & Destructor Documentation

6.544.2.1 **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::MessageIdMarshaller**  
 ( ) [inline]

6.544.2.2 **virtual activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::~~MessageIdMarshaller**  
 ( ) [inline, virtual]

### 6.544.3 Member Function Documentation

6.544.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.544.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.544 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference 2639

```
6.544.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.544.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.544.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.544.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.544.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageIdMarshaller.h**

## 6.545 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2632).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**:

## Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.545.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2632).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.545.2 Constructor & Destructor Documentation

6.545.2.1 **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.545.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.545.3 Member Function Documentation

6.545.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.545.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).



## 6.545 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference 2643

```
6.545.3.3 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.545.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.545.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.545.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.545.3.7 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## 6.546 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference 2645

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**MessageIdMarshaller.h**

## 6.546 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2636).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**:

### Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.546.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2636).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.546.2 Constructor & Destructor Documentation

6.546.2.1 **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.546.2.2 **virtual activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.546.3 Member Function Documentation

6.546.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.546.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.546 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference 2647

```
6.546.3.3 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.546.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.546.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.546.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.546.3.7 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MessageIdMarshaller.h**

## 6.547 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2640).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**:

## Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.547.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2640).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.547.2 Constructor & Destructor Documentation

6.547.2.1 **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.547.2.2 **virtual activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.547.3 Member Function Documentation

6.547.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.547.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).



## 6.547 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference 2651

```
6.547.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.547.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.547.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.547.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.547.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageIdMarshaller.h**

## 6.548 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2644).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller**:

## Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.548.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2644).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.548.2 Constructor & Destructor Documentation

6.548.2.1 **activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.548.2.2 **virtual activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.548.3 Member Function Documentation

6.548.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.548.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.548 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller Class Reference 2655

```
6.548.3.3 virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.548.3.4 virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.548.3.5 virtual int activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.548.3.6 virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.548.3.7 virtual void activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**MessageIdMarshaller.h**

## 6.549 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2648).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**:

## Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.549.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2648).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.549.2 Constructor & Destructor Documentation

6.549.2.1 **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::MessageIdMarshaller**  
( ) [inline]

6.549.2.2 **virtual activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::~~MessageIdMarshaller**  
( ) [inline, virtual]

### 6.549.3 Member Function Documentation

6.549.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.549.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).



## 6.549 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference 2659

```
6.549.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.549.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.549.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.549.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.549.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageIdMarshaller.h**

## 6.550 cms::MessageListener Class Reference

A **MessageListener** (p. 2652) object is used to receive asynchronously delivered messages.

```
#include <src/main/cms/MessageListener.h>
```

### Public Member Functions

- virtual **~MessageListener** ()
- virtual void **onMessage** (const **Message** \*message)=0  
*Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2493) types.*

#### 6.550.1 Detailed Description

A **MessageListener** (p. 2652) object is used to receive asynchronously delivered messages.

#### Since

1.0

#### 6.550.2 Constructor & Destructor Documentation

6.550.2.1 virtual cms::MessageListener::~~MessageListener ( ) [inline, virtual]

#### 6.550.3 Member Function Documentation

6.550.3.1 virtual void cms::MessageListener::onMessage ( const **Message** \* message )  
[pure virtual]

Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2493) types.

a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the onMessage function after this call-back returns

the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this **Message** (p. 2493).

It is considered a programming error for this method to throw an exception.

#### Parameters

<i>message</i>	<b>Message</b> (p. 2493) object {const} pointer recipient does not own.
----------------	---

The documentation for this class was generated from the following file:

- src/main/cms/**MessageListener.h**

## 6.551 activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2653).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageMarshaller:

#### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.551.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2653).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.551.2 Constructor & Destructor Documentation

6.551.2.1    `activemq::wireformat::openwire::marshal::v5::MessageMarshaller::MessageMarshaller`  
               ( ) [inline]

6.551.2.2    `virtual activemq::wireformat::openwire::marshal::v5::MessageMarshaller::~~MessageMarshaller`  
               ( ) [inline, virtual]

## 6.551.3 Member Function Documentation

6.551.3.1    `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseMarshal`  
               ( `OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut` ) throw ( `decaf::io::IOException` ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 195), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 234), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 358), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 385), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 430), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 537), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 649).

```
6.551.3.2 virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 196), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 234), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 358), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 385), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 431), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 537), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 650).

```
6.551.3.3 virtual int activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.551 activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference 2665

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 196), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 235), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 359), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 385), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 431), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 538), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 650).

```
6.551.3.4  virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal2
           ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
             decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
           ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 197), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 235), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 359), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 386), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 432), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 538), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 651).

```
6.551.3.5  virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightUnmarshal
           ( OpenWireFormat * wireFormat, commands::DataStructure *
             dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
             bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 197), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 236), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 360), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 386), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 432), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 539), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 651).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h`

## 6.552 **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2657).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**:

**Public Member Functions**

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )



*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.552.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2657).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.552.2 Constructor & Destructor Documentation

6.552.2.1 activemq::wireformat::openwire::marshal::v3::MessageMarshaller::MessageMarshaller  
( ) [inline]

6.552.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessageMarshaller::~~MessageMarshaller  
( ) [inline, virtual]

## 6.552.3 Member Function Documentation

6.552.3.1 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseMarshal  
( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 731).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 179), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 222), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 418), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 525), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 637).

```
6.552.3.2  virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 732).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 180), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 222), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 346), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 373), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 419), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 525), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 638).

```
6.552.3.3  virtual int activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

## 6.552 activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference 2669

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 180), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 347), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 373), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 419), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 526), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 638).

```
6.552.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 181), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 223), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

(p. 347), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 374), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 420), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 526), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 639).

```
6.552.3.5  virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 736).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 181), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 224), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 348), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 374), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 420), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 527), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 639).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h`

## 6.553 `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2661).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageMarshaller`:

## Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.553.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2661).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.553.2 Constructor & Destructor Documentation

6.553.2.1 **activemq::wireformat::openwire::marshal::v2::MessageMarshaller::MessageMarshaller**  
( ) [*inline*]

6.553.2.2 **virtual activemq::wireformat::openwire::marshal::v2::MessageMarshaller::~~MessageMarshaller**  
( ) [*inline, virtual*]

### 6.553.3 Member Function Documentation

```
6.553.3.1 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 191), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 242), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 362), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 434), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 541), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 657).

```
6.553.3.2 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

## 6.553 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference 2673

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 192), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 242), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 362), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 435), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 541), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 658).

```
6.553.3.3 virtual int activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 192), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 243), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 363), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 435), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 542), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 658).

```
6.553.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 193), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 243), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 363), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 390), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 436), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 542), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 659).

```
6.553.3.5 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 193), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 244), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 364), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 390), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**



## 6.554 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference 2675

(p. 436), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 543), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 659).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h`

## 6.554 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2666).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::MessageMarshaller`:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.554.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2666).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.554.2 Constructor & Destructor Documentation

6.554.2.1 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller::MessageMarshaller ( ) [inline]`

6.554.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageMarshaller::~~MessageMarshaller ( ) [inline, virtual]`

### 6.554.3 Member Function Documentation

6.554.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 738).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 187), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 230), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 381), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 426), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 533), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 641).

## 6.554 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference 2677

6.554.3.2 virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 188), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 230), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 354), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 381), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 427), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 533), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 642).

6.554.3.3 virtual int activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 740).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 355), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 381), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 427), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 534), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 642).

```
6.554.3.4  virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 741).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 189), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 231), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 355), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 382), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 428), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 534), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 643).

```
6.554.3.5  virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

## 6.555 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference 2679

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 232), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 356), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 382), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 428), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 535), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 643).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**MessageMarshaller.h**

## 6.555 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2670).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**:

### Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.555.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2670).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.555.2 Constructor & Destructor Documentation

6.555.2.1 **activemq::wireformat::openwire::marshal::v1::MessageMarshaller::MessageMarshaller**  
( ) [*inline*]

6.555.2.2 **virtual activemq::wireformat::openwire::marshal::v1::MessageMarshaller::~~MessageMarshaller**  
( ) [*inline, virtual*]

### 6.555.3 Member Function Documentation

6.555.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseMarshal**  
( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataOutputStream** \* *dataOut* ) throw ( **decaf::io::IOException** ) [*virtual*]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.555 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference 2681

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 226), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 350), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 422), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 529), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 645).

```
6.555.3.2  virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 226), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 350), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 529), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 646).

```
6.555.3.3  virtual int activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 377), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 530), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 646).

```
6.555.3.4  virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 227), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**



## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference

2683

(p. 351), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 378), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 424), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 530), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 647).

```
6.555.3.5 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 228), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 352), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 378), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 424), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 531), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 647).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageMarshaller.h**

## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2674).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**:

## Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.556.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2674).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.556.2 Constructor & Destructor Documentation

6.556.2.1 **activemq::wireformat::openwire::marshal::v6::MessageMarshaller::MessageMarshaller**  
 ( ) [inline]

6.556.2.2 **virtual activemq::wireformat::openwire::marshal::v6::MessageMarshaller::~~MessageMarshaller**  
 ( ) [inline, virtual]

### 6.556.3 Member Function Documentation

## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference 2685

6.556.3.1 `virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 199), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 238), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 366), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 438), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 545), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 653).

6.556.3.2 `virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 238), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 439), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 545), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 654).

```
6.556.3.3  virtual int activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from `activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller` (p. 760).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller` (p. 239), `activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller` (p. 367), `activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller` (p. 439), `activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller` (p. 546), and `activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller` (p. 654).

```
6.556.3.4  virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

## 6.556 activemq::wireformat::openwire::marshal::v6::MessageMarshaller Class Reference 2687

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 239), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 367), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller** (p. 440), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 546), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 655).

```
6.556.3.5 virtual void activemq::wireformat::openwire::marshal::v6::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller** (p. 201), **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller** (p. 240), **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller** (p. 368), **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**

(p. 440), **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller** (p. 547), and **activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller** (p. 655).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h`

## 6.557 cms::MessageNotReadableException Class Reference

This exception must be thrown when a CMS client attempts to read a write-only message.

```
#include <src/main/cms/MessageNotReadableException.h>
```

Inheritance diagram for cms::MessageNotReadableException:

### Public Member Functions

- **MessageNotReadableException** () throw ()
- **MessageNotReadableException** (const **MessageNotReadableException** &ex) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotReadableException** () throw ()

### 6.557.1 Detailed Description

This exception must be thrown when a CMS client attempts to read a write-only message.

#### Since

1.3

### 6.557.2 Constructor & Destructor Documentation

6.557.2.1 **cms::MessageNotReadableException::MessageNotReadableException** ( ) throw ()

6.557.2.2 **cms::MessageNotReadableException::MessageNotReadableException** ( const **MessageNotReadableException** & ex ) throw ()

- 6.557.2.3 `cms::MessageNotReadableException::MessageNotReadableException ( const std::string & message, const std::exception * cause ) throw ()`
- 6.557.2.4 `cms::MessageNotReadableException::MessageNotReadableException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`
- 6.557.2.5 `virtual cms::MessageNotReadableException::~~MessageNotReadableException ( ) throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotReadableException.h`

## 6.558 cms::MessageNotWriteableException Class Reference

This exception must be thrown when a CMS client attempts to write to a read-only message.

```
#include <src/main/cms/MessageNotWriteableException.h>
```

Inheritance diagram for cms::MessageNotWriteableException:

### Public Member Functions

- **MessageNotWriteableException** () throw ()
- **MessageNotWriteableException** (const **MessageNotWriteableException** &ex) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception \*cause) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotWriteableException** () throw ()

### 6.558.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since

1.3

## 6.558.2 Constructor & Destructor Documentation

- 6.558.2.1 `cms::MessageNotWriteableException::MessageNotWriteableException ( ) throw ()`
- 6.558.2.2 `cms::MessageNotWriteableException::MessageNotWriteableException ( const MessageNotWriteableException & ex ) throw ()`
- 6.558.2.3 `cms::MessageNotWriteableException::MessageNotWriteableException ( const std::string & message, const std::exception * cause ) throw ()`
- 6.558.2.4 `cms::MessageNotWriteableException::MessageNotWriteableException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`
- 6.558.2.5 `virtual cms::MessageNotWriteableException::~MessageNotWriteableException ( ) throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotWriteableException.h`

## 6.559 cms::MessageProducer Class Reference

A client uses a **MessageProducer** (p. 2681) object to send messages to a **Destination** (p. 1688).

```
#include <src/main/cms/MessageProducer.h>
```

Inheritance diagram for cms::MessageProducer:

### Public Member Functions

- virtual `~MessageProducer ()`
- virtual void **send** (**Message** \*message)=0 throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (**Message** \*message, int deliveryMode, int priority, long long timeToLive)=0 throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )  
*Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.*
- virtual void **send** (const **Destination** \*destination, **Message** \*message)=0 throw ( cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )



*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*

- virtual void **send** (const **Destination** \*destination, **Message** \*message, int deliveryMode, int priority, long long timeToLive)=0 throw ( cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException )

*Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.*

- virtual void **setDeliveryMode** (int mode)=0 throw ( CMSEException )

*Sets the delivery mode for this Producer.*

- virtual int **getDeliveryMode** () const =0 throw ( CMSEException )

*Gets the delivery mode for this Producer.*

- virtual void **setDisableMessageID** (bool value)=0 throw ( CMSEException )

*Sets if **Message** (p. 2493) IDs are disabled for this Producer.*

- virtual bool **getDisableMessageID** () const =0 throw ( CMSEException )

*Gets if **Message** (p. 2493) IDs are disabled for this Producer.*

- virtual void **setDisableMessageTimeStamp** (bool value)=0 throw ( CMSEException )

*Sets if **Message** (p. 2493) Time Stamps are disabled for this Producer.*

- virtual bool **getDisableMessageTimeStamp** () const =0 throw ( CMSEException )

*Gets if **Message** (p. 2493) Time Stamps are disabled for this Producer.*

- virtual void **setPriority** (int priority)=0 throw ( CMSEException )

*Sets the Priority that this Producers sends messages at.*

- virtual int **getPriority** () const =0 throw ( CMSEException )

*Gets the Priority level that this producer sends messages at.*

- virtual void **setTimeToLive** (long long time)=0 throw ( CMSEException )

*Sets the Time to Live that this Producers sends messages with.*

- virtual long long **getTimeToLive** () const =0 throw ( CMSEException )

*Gets the Time to Live that this producer sends messages with.*

### 6.559.1 Detailed Description

A client uses a **MessageProducer** (p. 2681) object to send messages to a **Destination** (p. 1688).

A **MessageProducer** (p. 2681) object is created by passing a **Destination** (p. 1688) object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p. 1688) must be provided with every send operation. A typical use for this kind of message producer is to send replies to requests using the request's CMSReplyTo destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

#### Since

1.0

### 6.559.2 Constructor & Destructor Documentation

6.559.2.1 `virtual cms::MessageProducer::~MessageProducer ( ) [inline, virtual]`

### 6.559.3 Member Function Documentation

6.559.3.1 `virtual int cms::MessageProducer::getDeliveryMode ( ) const throw ( CMSException ) [pure virtual]`

Gets the delivery mode for this Producer.

#### Returns

The **DeliveryMode** (p. 1687)

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs.
----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1046), and **activemq::core::ActiveMQProducer** (p. 443).

6.559.3.2 `virtual bool cms::MessageProducer::getDisableMessageID ( ) const throw ( CMSException ) [pure virtual]`

Gets if **Message** (p. 2493) IDs are disabled for this Producer.

#### Returns

boolean indicating enable / disable (true / false)

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs.
----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1046), and **activemq::core::ActiveMQProducer** (p. 443).

6.559.3.3 `virtual bool cms::MessageProducer::getDisableMessageTimeStamp ( ) const throw ( CMSEException ) [pure virtual]`

Gets if **Message** (p. 2493) Time Stamps are disabled for this Producer.

#### Returns

boolean indicating enable / disable (true / false)

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1047), and **activemq::core::ActiveMQProducer** (p. 443).

6.559.3.4 `virtual int cms::MessageProducer::getPriority ( ) const throw ( CMSEException ) [pure virtual]`

Gets the Priority level that this producer sends messages at.

#### Returns

int based priority level

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1047), and **activemq::core::ActiveMQProducer** (p. 444).

6.559.3.5 `virtual long long cms::MessageProducer::getTimeToLive ( ) const throw ( CMSEException ) [pure virtual]`

Gets the Time to Live that this producer sends messages with.

#### Returns

Time to live value in milliseconds

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1047), and **activemq::core::ActiveMQProducer** (p. 444).

```
6.559.3.6 virtual void cms::MessageProducer::send ( Message * message, int
           deliveryMode, int priority, long long timeToLive ) throw ( cms::CMSEException,
           cms::MessageFormatException, cms::InvalidDestinationException,
           cms::UnsupportedOperationException ) [pure virtual]
```

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>message</i>	The message to be sent.
<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs while sending the message.
<b>MessageFormatException</b> (p. 2622)	- if an Invalid <b>Message</b> (p. 2493) is given.
<b>InvalidDestinationException</b> (p. 2093)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) with an invalid destination.
<b>UnsupportedOperationException</b> (p. 3852)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1048), and **activemq::core::ActiveMQProducer** (p. 446).

```
6.559.3.7 virtual void cms::MessageProducer::send ( const Destination
           * destination, Message * message, int deliveryMode, int
           priority, long long timeToLive ) throw ( cms::CMSEException,
           cms::MessageFormatException, cms::InvalidDestinationException,
           cms::UnsupportedOperationException ) [pure virtual]
```

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	The message to be sent.

<i>delivery-Mode</i>	The delivery mode to be used.
<i>priority</i>	The priority for this message.
<i>timeToLive</i>	The time to live value for this message in milliseconds.

### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs while sending the message.
<b>MessageFormatException</b> (p. 2622)	- if an Invalid <b>Message</b> (p. 2493) is given.
<b>InvalidDestinationException</b> (p. 2093)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) with an invalid destination.
<b>UnsupportedOperationException</b> (p. 3852)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1049), and **activemq::core::ActiveMQProducer** (p. 445).

```
6.559.3.8 virtual void cms::MessageProducer::send ( Message
* message ) throw ( cms::CMSException,
cms::MessageFormatException, cms::InvalidDestinationException,
cms::UnsupportedOperationException ) [pure virtual]
```

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

### Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs while sending the message.
<b>MessageFormatException</b> (p. 2622)	- if an Invalid <b>Message</b> (p. 2493) is given.
<b>InvalidDestinationException</b> (p. 2093)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) with an invalid destination.
<b>UnsupportedOperationException</b> (p. 3852)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1049), and **activemq::core::ActiveMQProducer** (p. 446).

6.559.3.9 `virtual void cms::MessageProducer::send ( const Destination *  
destination, Message * message ) throw ( cms::CMSEException,  
cms::MessageFormatException, cms::InvalidDestinationException,  
cms::UnsupportedOperationException ) [pure virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

#### Parameters

<i>destination</i>	The destination on which to send the message
<i>message</i>	the message to be sent.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs while sending the message.
<b>MessageFormatException</b> (p. 2622)	- if an Invalid <b>Message</b> (p. 2493) is given.
<b>InvalidDestinationException</b> (p. 2093)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) with an invalid destination.
<b>UnsupportedOperationException</b> (p. 3852)	- if a client uses this method with a <b>MessageProducer</b> (p. 2681) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 1048), and **activemq::core::ActiveMQProducer** (p. 447).

6.559.3.10 `virtual void cms::MessageProducer::setDeliveryMode ( int mode ) throw ( CMSEException ) [pure virtual]`

Sets the delivery mode for this Producer.

#### Parameters

<i>mode</i>	The <b>DeliveryMode</b> (p. 1687)
-------------	-----------------------------------

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1050), and **activemq::core::ActiveMQProducer** (p. 447).

6.559.3.11 `virtual void cms::MessageProducer::setDisableMessageID ( bool value ) throw ( CMSEException ) [pure virtual]`

Sets if **Message** (p. 2493) Ids are disabled for this Producer.

#### Parameters

<i>value</i>	boolean indicating enable / disable (true / false)
--------------	--

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1050), and **activemq::core::ActiveMQProducer** (p. 447).

6.559.3.12 `virtual void cms::MessageProducer::setDisableMessageTimeStamp ( bool value ) throw ( CMSEException ) [pure virtual]`

Sets if **Message** (p. 2493) Time Stamps are disabled for this Producer.

#### Parameters

<i>value</i>	- boolean indicating enable / disable (true / false)
--------------	--

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1050), and **activemq::core::ActiveMQProducer** (p. 448).

6.559.3.13 `virtual void cms::MessageProducer::setPriority ( int priority ) throw ( CMSEException ) [pure virtual]`

Sets the Priority that this Producers sends messages at.

#### Parameters

<i>priority</i>	int value for Priority level
-----------------	------------------------------

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1051), and **activemq::core::ActiveMQProducer** (p. 448).

6.559.3.14 `virtual void cms::MessageProducer::setTimeToLive ( long long time ) throw ( CMSEException ) [pure virtual]`

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

#### Parameters

<i>time</i>	default time to live value in milliseconds
-------------	--

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::CachedProducer** (p. 1051), and **activemq::core::ActiveMQProducer** (p. 448).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageProducer.h`

## 6.560 **activemq::wireformat::openwire::utils::MessagePropertyInterceptor** Class Reference

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

```
#include <src/main/activemq/wireformat/openwire/utils/MessagePropertyInter
```

#### Public Member Functions

- **MessagePropertyInterceptor** (**commands::Message** \*message, **util::PrimitiveMap** \*properties) throw ( **decaf::lang::exceptions::NullPointerException** )  
*Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.*
- virtual **~MessagePropertyInterceptor** ()
- virtual bool **getBooleanProperty** (const std::string &name) const  
*Gets a boolean property.*
- virtual unsigned char **getByteProperty** (const std::string &name) const  
*Gets a byte property.*
- virtual double **getDoubleProperty** (const std::string &name) const



*Gets a double property.*

- virtual float **getFloatProperty** (const std::string &name) const

*Gets a float property.*

- virtual int **getIntProperty** (const std::string &name) const

*Gets a int property.*

- virtual long long **getLongProperty** (const std::string &name) const

*Gets a long property.*

- virtual short **getShortProperty** (const std::string &name) const

*Gets a short property.*

- virtual std::string **getStringProperty** (const std::string &name) const

*Gets a string property.*

- virtual void **setBooleanProperty** (const std::string &name, bool value)

*Sets a boolean property.*

- virtual void **setByteProperty** (const std::string &name, unsigned char value)

*Sets a byte property.*

- virtual void **setDoubleProperty** (const std::string &name, double value)

*Sets a double property.*

- virtual void **setFloatProperty** (const std::string &name, float value)

*Sets a float property.*

- virtual void **setIntProperty** (const std::string &name, int value)

*Sets a int property.*

- virtual void **setLongProperty** (const std::string &name, long long value)

*Sets a long property.*

- virtual void **setShortProperty** (const std::string &name, short value)

*Sets a short property.*

- virtual void **setStringProperty** (const std::string &name, const std::string &value)

*Sets a string property.*

### 6.560.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

Currently the only properties that are intercepted and handled are:

Name		Conversion Supported	-----	JMSXDeliveryCount
Int, Long, String		JMSXGroupID		String JMSXGroupSeq
		Int, Long, String		

## 6.560.2 Constructor & Destructor Documentation

6.560.2.1 `activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInterceptor ( commands::Message * message, util::PrimitiveMap * properties ) throw ( decaf::lang::exceptions::NullPointerException )`

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

### Parameters

<i>message</i>	- The Message to store reserved property data in
<i>properties</i>	- The PrimitiveMap to store the rest of the properties in.

### Exceptions

<i>NullPointerException</i>	if either param is NULL
-----------------------------	-------------------------

6.560.2.2 `virtual activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~~MessagePropertyInterceptor ( ) [virtual]`

## 6.560.3 Member Function Documentation

6.560.3.1 `virtual bool activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty ( const std::string & name ) const [virtual]`

Gets a boolean property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

### Returns

The value for the named property.

6.560.3.2 `virtual unsigned char activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getByteProperty ( const std::string & name ) const [virtual]`

Gets a byte property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

### Returns

The value for the named property.

## 6.560 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference 2701

6.560.3.3 virtual double activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty  
( const std::string & *name* ) const [virtual]

Gets a double property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

### Returns

The value for the named property.

6.560.3.4 virtual float activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty  
( const std::string & *name* ) const [virtual]

Gets a float property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

### Returns

The value for the named property.

6.560.3.5 virtual int activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getIntProperty  
( const std::string & *name* ) const [virtual]

Gets a int property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

### Returns

The value for the named property.

6.560.3.6 virtual long long activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getLongProperty  
( const std::string & *name* ) const [virtual]

Gets a long property.

### Parameters

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

```
6.560.3.7  virtual short activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getShortProperty  
           ( const std::string & name ) const    [virtual]
```

Gets a short property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

```
6.560.3.8  virtual std::string activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty  
           ( const std::string & name ) const    [virtual]
```

Gets a string property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
-------------	---------------------------------------

**Returns**

The value for the named property.

```
6.560.3.9  virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty  
           ( const std::string & name, bool value ) [virtual]
```

Sets a boolean property.

**Parameters**

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

```
6.560.3.10 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty  
           ( const std::string & name, unsigned char value ) [virtual]
```

Sets a byte property.

## 6.560 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference 2703

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.560.3.11 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty ( const std::string & name, double value ) [virtual]`

Sets a double property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.560.3.12 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setFloatProperty ( const std::string & name, float value ) [virtual]`

Sets a float property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.560.3.13 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setIntProperty ( const std::string & name, int value ) [virtual]`

Sets a int property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.560.3.14 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setLongProperty ( const std::string & name, long long value ) [virtual]`

Sets a long property.

### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.560.3.15 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setShortProperty ( const std::string & name, short value ) [virtual]`

Sets a short property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

6.560.3.16 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setStringProperty ( const std::string & name, const std::string & value ) [virtual]`

Sets a string property.

#### Parameters

<i>name</i>	The name of the property to retrieve.
<i>value</i>	The value for the named property.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h`

## 6.561 activemq::commands::MessagePull Class Reference

```
#include <src/main/activemq/commands/MessagePull.h>
```

Inheritance diagram for `activemq::commands::MessagePull`:

### Public Member Functions

- **MessagePull** ()
- virtual **~MessagePull** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **MessagePull** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &correlationId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
 throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_MESSAGEPULL** = 20

### Protected Attributes

- **Pointer**< **ConsumerId** > consumerId
- **Pointer**< **ActiveMQDestination** > destination
- long long timeout
- std::string correlationId
- **Pointer**< **MessageId** > messageId

## 6.561.1 Constructor & Destructor Documentation

6.561.1.1 activemq::commands::MessagePull::MessagePull ( )

6.561.1.2 virtual activemq::commands::MessagePull::~~MessagePull ( ) [virtual]

## 6.561.2 Member Function Documentation

6.561.2.1 **virtual MessagePull\*** activemq::commands::MessagePull::cloneDataStructure ( )  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.561.2.2 **virtual void** activemq::commands::MessagePull::copyDataStructure ( **const DataStructure \* src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.561.2.3 **virtual bool** activemq::commands::MessagePull::equals ( **const DataStructure \* value** ) **const** [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.561.2.4 **virtual const Pointer<ConsumerId>&** activemq::commands::MessagePull::getConsumerId ( ) **const**  
[virtual]

6.561.2.5 **virtual Pointer<ConsumerId>&** activemq::commands::MessagePull::getConsumerId ( )  
[virtual]

6.561.2.6 **virtual const std::string&** activemq::commands::MessagePull::getCorrelationId ( )  
**const** [virtual]



6.561.2.7 virtual std::string& activemq::commands::MessagePull::getCorrelationId ( )  
[virtual]

6.561.2.8 virtual unsigned char activemq::commands::MessagePull::getDataStructureType ( )  
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.561.2.9 virtual const Pointer<ActiveMQDestination>&  
activemq::commands::MessagePull::getDestination ( ) const [virtual]

6.561.2.10 virtual Pointer<ActiveMQDestination>&  
activemq::commands::MessagePull::getDestination ( ) [virtual]

6.561.2.11 virtual const Pointer<MessageId>& ac-  
tivemq::commands::MessagePull::getMessageId ( ) const  
[virtual]

6.561.2.12 virtual Pointer<MessageId>& ac-  
tivemq::commands::MessagePull::getMessageId ( )  
[virtual]

6.561.2.13 virtual long long activemq::commands::MessagePull::getTimeout ( ) const  
[virtual]

6.561.2.14 virtual void activemq::commands::MessagePull::setConsumerId ( const Pointer<  
**ConsumerId**> & *consumerId* ) [virtual]

6.561.2.15 virtual void activemq::commands::MessagePull::setCorrelationId ( const std::string  
& *correlationId* ) [virtual]

6.561.2.16 virtual void activemq::commands::MessagePull::setDestination ( const Pointer<  
**ActiveMQDestination**> & *destination* ) [virtual]

6.561.2.17 virtual void activemq::commands::MessagePull::setMessageId ( const Pointer<  
**MessageId**> & *messageId* ) [virtual]

6.561.2.18 virtual void activemq::commands::MessagePull::setTimeout ( long long *timeout* )  
[virtual]

6.561.2.19 `virtual std::string activemq::commands::MessagePull::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.561.2.20 `virtual Pointer<Command> activemq::commands::MessagePull::visit`  
`( activemq::state::CommandVisitor * visitor ) throw (`  
`exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.561.3 Field Documentation

6.561.3.1 `Pointer<ConsumerId> activemq::commands::MessagePull::consumerId`  
`[protected]`

6.561.3.2 `std::string activemq::commands::MessagePull::correlationId`  
`[protected]`

6.561.3.3 `Pointer<ActiveMQDestination> ac-`  
`tivemq::commands::MessagePull::destination`  
`[protected]`

6.561.3.4 `const unsigned char activemq::commands::MessagePull::ID_ -`  
`MESSAGEPULL = 20 [static]`

6.561.3.5 `Pointer<Messageld> activemq::commands::MessagePull::messageld`  
`[protected]`

6.561.3.6 `long long activemq::commands::MessagePull::timeout [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessagePull.h`

## 6.562 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2700).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.562.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2700).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.562.2 Constructor & Destructor Documentation

6.562.2.1 `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::MessagePullMarshaller ( ) [inline]`

6.562.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::~~MessagePullMarshaller ( ) [inline, virtual]`

## 6.562.3 Member Function Documentation

6.562.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.562.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.562.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.562.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

6.562.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.562.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.562.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessagePullMarshaller.h**

## 6.563 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2704).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.563.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2704).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.563.2 Constructor & Destructor Documentation

6.563.2.1 `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::MessagePullMarshaller ( ) [inline]`

6.563.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::~~MessagePullMarshaller ( ) [inline, virtual]`

### 6.563.3 Member Function Documentation

6.563.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.563.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.563.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.563.3.4  virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.563.3.5  virtual int activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.563.3.6  virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.563.3.7  virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**MessagePullMarshaller.h**

## 6.564 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2708).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.564.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2708).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.564.2 Constructor & Destructor Documentation

6.564.2.1 `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::MessagePullMarshaller ( ) [inline]`

6.564.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::~~MessagePullMarshaller ( ) [inline, virtual]`

## 6.564.3 Member Function Documentation

6.564.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.564.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.564.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.564.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.564.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.564.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.564.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessagePullMarshaller.h**

## 6.565 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2712).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.565.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2712).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.565.2 Constructor & Destructor Documentation

6.565.2.1 `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::MessagePullMarshaller ( )` `[inline]`

6.565.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::~~MessagePullMarshaller ( )` `[inline, virtual]`

## 6.565.3 Member Function Documentation

6.565.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::createObject ( )`  
`const` `[virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.565.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::getDataStructureType ( )` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.565.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut )` `throw ( decaf::io::IOException )` `[virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.565.3.4  virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.565.3.5  virtual int activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.565.3.6  virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.565.3.7  virtual void activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**MessagePullMarshaller.h**

## 6.566 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2716).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.566.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2716).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.566.2 Constructor & Destructor Documentation

6.566.2.1 `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::MessagePullMarshaller ( ) [inline]`

6.566.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::~~MessagePullMarshaller ( ) [inline, virtual]`

## 6.566.3 Member Function Documentation

6.566.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.566.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.566.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.566.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

6.566.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.566.3.6  virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.566.3.7  virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessagePullMarshaller.h**

## 6.567 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2720).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller:

### Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.567.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2720).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.567.2 Constructor & Destructor Documentation

6.567.2.1 `activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::MessagePullMarshaller ( ) [inline]`

6.567.2.2 `virtual activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::~~MessagePullMarshaller ( ) [inline, virtual]`

## 6.567.3 Member Function Documentation

6.567.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.567.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.567.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.567.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.567.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.567.3.6  virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.567.3.7  virtual void activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**MessagePullMarshaller.h**

## 6.568 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p.2724) defines a base level **Transport** (p.3819) class that is intended to be used in place of an a regular protocol **Transport** (p.3819) such as TCP.

```
#include <src/main/activemq/transport/mock/MockTransport.h>
```

Inheritance diagram for activemq::transport::mock::MockTransport:

### Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual ~**MockTransport** ()
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
 

*Sets the **ResponseBuilder** (p. 3231) that this class uses to create Responses to Commands sent.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
 

*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
 

*Sends the given command to the broker and then waits for the response.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
 

*Sends the given command to the broker and then waits for the response.*
- virtual void **setOutgoingListener** (**TransportListener** \*listener)
 

*Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP\_UNUSED)
 

*Sets the WireFormat instance to use.*
- **Pointer**< **wireformat::WireFormat** > **getWireFormat** () const
 

*Gets the currently set WireFormat.*
- virtual void **setTransportListener** (**TransportListener** \*listener)
 

*Sets the observer of asynchronous exceptions from this transport.*
- virtual **TransportListener** \* **getTransportListener** () const
 

*Gets the observer of asynchronous exceptions from this transport.*
- virtual void **fireCommand** (const **Pointer**< **Command** > &command)
 

*Fires a Command back through this transport to its registered CommandListener if there is one.*

- virtual void **fireException** (const **exceptions::ActiveMQException** &ex)  
*Fires a Exception back through this transport to its registered ExceptionListener if there is one.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts the **Transport** (p. 3819), the send methods of a **Transport** (p. 3819) will throw an exception if used before the **Transport** (p. 3819) is started.*
- virtual void **stop** () throw ( decaf::io::IOException )  
*Stops the **Transport** (p. 3819).*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeid)  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3819) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3819) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const decaf::net::URI &uri AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*reconnect to another location*
- bool **isFailOnSendMessage** () const
- void **setFailOnSendMessage** (bool value)
- int **getNumSendMessageBeforeFail** () const
- void **setNumSendMessageBeforeFail** (int value)
- int **getNumSentMessages** () const
- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)
- bool **isFailOnKeepAliveSends** () const
- void **setFailOnKeepAliveSends** (bool value)
- int **getNumSentKeepAlivesBeforeFail** () const
- void **setNumSentKeepAlivesBeforeFail** (int value)
- int **getNumSentKeepAlives** () const
- void **setNumSentKeepAlives** (int value)
- bool **isFailOnStart** () const
- void **setFailOnStart** (bool value)
- bool **isFailOnStop** () const
- void **setFailOnStop** (bool value)
- bool **isFailOnClose** () const
- void **setFailOnClose** (bool value)

## Static Public Member Functions

- static **MockTransport** \* **getInstance** ()

### 6.568.1 Detailed Description

The **MockTransport** (p. 2724) defines a base level **Transport** (p. 3819) class that is intended to be used in place of an a regular protocol **Transport** (p. 3819) such as TCP.

This **Transport** (p. 3819) assumes that it is the base **Transport** (p. 3819) in the Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p. 3819) defines an Interface **ResponseBuilder** (p. 3231) which must be implemented by any protocol for which the **Transport** (p. 3819) is used to Emulate. The **Transport** (p. 3819) hands off all outbound commands to the **ResponseBuilder** (p. 3231) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

### 6.568.2 Constructor & Destructor Documentation

6.568.2.1 `activemq::transport::mock::MockTransport::MockTransport ( const Pointer< wireformat::WireFormat > & wireFormat, const Pointer< ResponseBuilder > & responseBuilder )`

6.568.2.2 `virtual activemq::transport::mock::MockTransport::~MockTransport ( ) [inline, virtual]`

### 6.568.3 Member Function Documentation

6.568.3.1 `virtual void activemq::transport::mock::MockTransport::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<i>IOException</i>	if an error occurs while closing.
--------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 1121).

6.568.3.2 `virtual void activemq::transport::mock::MockTransport::fireCommand ( const Pointer< Command > & command ) [inline, virtual]`

Fires a Command back through this transport to its registered CommandListener if there is one.

**Parameters**

<i>command</i>	- Command to send to the Listener.
----------------	------------------------------------

6.568.3.3 `virtual void activemq::transport::mock::MockTransport::fireException ( const exceptions::ActiveMQException & ex ) [inline, virtual]`

Fires a Exception back through this transport to its registered ExceptionListener if there is one.

**Parameters**

<i>ex</i>	The Exception that will be passed on the the <b>Transport</b> (p. 3819) listener.
-----------	---

6.568.3.4 `static MockTransport* activemq::transport::mock::MockTransport::getInstance ( ) [inline, static]`

6.568.3.5 `int activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail ( ) const [inline]`

6.568.3.6 `int activemq::transport::mock::MockTransport::getNumReceivedMessages ( ) const [inline]`

6.568.3.7 `int activemq::transport::mock::MockTransport::getNumSentKeepAlives ( ) const [inline]`

6.568.3.8 `int activemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail ( ) const [inline]`

6.568.3.9 `int activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail ( ) const [inline]`

6.568.3.10 `int activemq::transport::mock::MockTransport::getNumSentMessages ( ) const [inline]`

6.568.3.11 `virtual std::string activemq::transport::mock::MockTransport::getRemoteAddress ( ) const [inline, virtual]`

**Returns**

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3821).

6.568.3.12 `virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener ( ) const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

#### Returns

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3821).

6.568.3.13 `Pointer<wireformat::WireFormat> activemq::transport::mock::MockTransport::getWireFormat ( ) const [inline]`

Gets the currently set WireFormat.

#### Returns

the current WireFormat object.

6.568.3.14 `virtual bool activemq::transport::mock::MockTransport::isClosed ( ) const [inline, virtual]`

Has the **Transport** (p. 3819) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3819)

Implements **activemq::transport::Transport** (p. 3821).

6.568.3.15 `virtual bool activemq::transport::mock::MockTransport::isConnected ( ) const [inline, virtual]`

Is the **Transport** (p. 3819) Connected to its Broker.

#### Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3821).

- 6.568.3.16 `bool activemq::transport::mock::MockTransport::isFailOnClose ( ) const`  
[inline]
- 6.568.3.17 `bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends ( ) const`  
[inline]
- 6.568.3.18 `bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage ( ) const`  
[inline]
- 6.568.3.19 `bool activemq::transport::mock::MockTransport::isFailOnSendMessage ( ) const`  
[inline]
- 6.568.3.20 `bool activemq::transport::mock::MockTransport::isFailOnStart ( ) const`  
[inline]
- 6.568.3.21 `bool activemq::transport::mock::MockTransport::isFailOnStop ( ) const`  
[inline]
- 6.568.3.22 `virtual bool activemq::transport::mock::MockTransport::isFaultTolerant ( ) const`  
[inline, virtual]

Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns

true if the **Transport** (p. 3819) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3822).

- 6.568.3.23 `virtual Transport* activemq::transport::mock::MockTransport::narrow ( const std::type_info & typeid )` [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.

#### Parameters

<i>typeid</i>	- The <code>type_info</code> of the Object we are searching for.
---------------	--

#### Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3822).



6.568.3.24 `virtual void activemq::transport::mock::MockTransport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
[virtual]

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3822).

6.568.3.25 `virtual void activemq::transport::mock::MockTransport::reconnect ( const decaf::net::URI &uri AMQCPP_UNUSED ) throw ( decaf::io::IOException )`  
[inline, virtual]

reconnect to another location

#### Parameters

<i>uri</i>	
------------	--

#### Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

6.568.3.26 `virtual Pointer<Response> activemq::transport::mock::MockTransport::request ( const Pointer< Command > & command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )`  
[virtual]

Sends the given command to the broker and then waits for the response.

#### Parameters

<i>command</i>	- The command to be sent.
<i>timeout</i>	- The time to wait for this response.

#### Returns

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3824).

6.568.3.27 **virtual Pointer<Response> activemq::transport::mock::MockTransport::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)**  
[virtual]

Sends the given command to the broker and then waits for the response.

**Parameters**

<i>command</i>	the command to be sent.
----------------	-------------------------

**Returns**

the response from the broker.

**Exceptions**

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3823).

6.568.3.28 **void activemq::transport::mock::MockTransport::setFailOnClose ( bool value )**  
[inline]

6.568.3.29 **void activemq::transport::mock::MockTransport::setFailOnKeepAliveSends ( bool value )** [inline]

6.568.3.30 **void activemq::transport::mock::MockTransport::setFailOnReceiveMessage ( bool value )** [inline]

6.568.3.31 **void activemq::transport::mock::MockTransport::setFailOnSendMessage ( bool value )** [inline]

6.568.3.32 **void activemq::transport::mock::MockTransport::setFailOnStart ( bool value )**  
[inline]

6.568.3.33 **void activemq::transport::mock::MockTransport::setFailOnStop ( bool value )**  
[inline]

- 6.568.3.34 `void activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail ( int value ) [inline]`
- 6.568.3.35 `void activemq::transport::mock::MockTransport::setNumReceivedMessages ( int value ) [inline]`
- 6.568.3.36 `void activemq::transport::mock::MockTransport::setNumSentKeepAlives ( int value ) [inline]`
- 6.568.3.37 `void activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail ( int value ) [inline]`
- 6.568.3.38 `void activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail ( int value ) [inline]`
- 6.568.3.39 `void activemq::transport::mock::MockTransport::setNumSentMessages ( int value ) [inline]`
- 6.568.3.40 `virtual void activemq::transport::mock::MockTransport::setOutgoingListener ( TransportListener * listener ) [inline, virtual]`

Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.

#### Parameters

<i>listener</i>	- The CommandListener to notify for each message
-----------------	--

- 6.568.3.41 `void activemq::transport::mock::MockTransport::setResponseBuilder ( const Pointer< ResponseBuilder > & responseBuilder ) [inline]`

Sets the **ResponseBuilder** (p. 3231) that this class uses to create Responses to Commands sent.

These are either real Response Objects, or Commands that would have been sent Asynchronously be the Broker.

#### Parameters

<i>response-Builder</i>	- The <b>ResponseBuilder</b> (p. 3231) to use from now on.
-------------------------	--

- 6.568.3.42 `virtual void activemq::transport::mock::MockTransport::setTransportListener ( TransportListener * listener ) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

**Parameters**

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 3824).

```
6.568.3.43  virtual void activemq::transport::mock::MockTransport::setWireFormat ( const
            Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED )
            [inline, virtual]
```

Sets the WireFormat instance to use.

**Parameters**

<i>wireFormat</i>	WireFormat the object used to encode / decode commands.
-------------------	---

```
6.568.3.44  virtual void activemq::transport::mock::MockTransport::start ( ) throw (
            decaf::io::IOException ) [virtual]
```

Starts the **Transport** (p. 3819), the send methods of a **Transport** (p. 3819) will throw an exception if used before the **Transport** (p. 3819) is started.

**Exceptions**

<i>IOException</i>	if an error occurs while starting the <b>Transport</b> (p. 3819).
--------------------	---

Implements **activemq::transport::Transport** (p. 3825).

```
6.568.3.45  virtual void activemq::transport::mock::MockTransport::stop ( ) throw (
            decaf::io::IOException ) [virtual]
```

Stops the **Transport** (p. 3819).

**Exceptions**

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implements **activemq::transport::Transport** (p. 3825).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransport.h**

## 6.569 activemq::transport::mock::MockTransportFactory Class Reference

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

```
#include <src/main/activemq/transport/mock/MockTransportFactory.h>
```

Inheritance diagram for activemq::transport::mock::MockTransportFactory:

### Public Member Functions

- virtual **~MockTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw ( exceptions::ActiveMQException )  
*Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw ( exceptions::ActiveMQException )  
*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

### Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw ( exceptions::ActiveMQException )  
*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

#### 6.569.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

#### 6.569.2 Constructor & Destructor Documentation

6.569.2.1 virtual activemq::transport::mock::MockTransportFactory::~MockTransportFactory ( ) [inline, virtual]

#### 6.569.3 Member Function Documentation

```
6.569.3.1 virtual Pointer<Transport> ac-
          tivemq::transport::mock::MockTransportFactory::create ( const
          decaf::net::URI & location ) throw ( exceptions::ActiveMQException )
          [virtual]
```

Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 3826).

```
6.569.3.2 virtual Pointer<Transport> ac-
          tivemq::transport::mock::MockTransportFactory::createComposite ( const
          decaf::net::URI & location ) throw ( exceptions::ActiveMQException )
          [virtual]
```

Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 3827).

```
6.569.3.3 virtual Pointer<Transport> ac-
          tivemq::transport::mock::MockTransportFactory::doCreateComposite ( const
          decaf::net::URI & location, const Pointer< wireformat::WireFormat
          > & wireFormat, const decaf::util::Properties & properties ) throw (
          exceptions::ActiveMQException ) [protected, virtual]
```

Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to.
<i>wireFormat</i>	- the assigned WireFormat for the new <b>Transport</b> (p. 3819).
<i>properties</i>	- Properties to apply to the transport.

**Returns**

Pointer to a new **Transport** (p. 3819) instance.

**Exceptions**

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransportFactory.h**

**6.570 decaf::util::concurrent::Mutex Class Reference**

**Mutex** (p. 2736) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

```
#include <src/main/decaf/util/concurrent/Mutex.h>
```

Inheritance diagram for decaf::util::concurrent::Mutex:

**Public Member Functions**

- **Mutex** ()
- virtual ~**Mutex** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to **Lock** (p. 2334) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.570.1 Detailed Description

**Mutex** (p. 2736) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

#### Since

1.0

### 6.570.2 Constructor & Destructor Documentation

6.570.2.1 decaf::util::concurrent::Mutex::Mutex ( )

6.570.2.2 virtual decaf::util::concurrent::Mutex::~~Mutex ( ) [virtual]

### 6.570.3 Member Function Documentation

6.570.3.1 virtual void decaf::util::concurrent::Mutex::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::lock(), decaf::util::StlMap< std::string, cms::Topic \* >::lock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock(), and decaf::util::AbstractCollection< cms::Connection \* >::lock().

6.570.3.2 virtual void decaf::util::concurrent::Mutex::notify ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions



<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notify(), decaf::util::StlMap< std::string, cms::Topic \* >::notify(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify(), and decaf::util::AbstractCollection< cms::Connection \* >::notify().

```
6.570.3.3 virtual void decaf::util::concurrent::Mutex::notifyAll ( )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notifyAll(), decaf::util::StlMap< std::string, cms::Topic \* >::notifyAll(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll(), and decaf::util::AbstractCollection< cms::Connection \* >::notifyAll().

```
6.570.3.4 virtual bool decaf::util::concurrent::Mutex::tryLock ( ) throw (
            decaf::lang::exceptions::RuntimeException ) [virtual]
```

Attempts to **Lock** (p. 2334) the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::tryLock()`, `decaf::util::StlMap< std::string, cms::Topic * >::tryLock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock()`, and `decaf::util::AbstractCollection< cms::Connection * >::tryLock()`.

6.570.3.5 `virtual void decaf::util::concurrent::Mutex::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException )` [virtual]

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::unlock()`, `decaf::util::StlMap< std::string, cms::Topic * >::unlock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock()`, and `decaf::util::AbstractCollection< cms::Connection * >::unlock()`.

6.570.3.6 `virtual void decaf::util::concurrent::Mutex::wait ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )` [virtual]

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::wait()`, `decaf::util::StlMap< std::string, cms::Topic * >::wait()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait()`, and `decaf::util::AbstractCollection< cms::Connection * >::wait()`.

6.570.3.7 virtual void decaf::util::concurrent::Mutex::wait ( long long *millisecs* ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

6.570.3.8 virtual void decaf::util::concurrent::Mutex::wait ( long long *millisecs*, int *nanos* ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.

<i>IllegalMonitorState-Exception</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.
--------------------------------------	---

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Mutex.h**

## 6.571 decaf::util::concurrent::MutexHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/MutexHandle.h>
```

### Public Member Functions

- **MutexHandle** ()
- **~MutexHandle** ()
- **MutexHandle** ()
- **~MutexHandle** ()

### Data Fields

- pthread\_mutex\_t **mutex**
- volatile long long **lock\_owner**
- volatile long long **lock\_count**
- CRITICAL\_SECTION **mutex**

### 6.571.1 Constructor & Destructor Documentation

6.571.1.1 decaf::util::concurrent::MutexHandle::MutexHandle ( ) [inline]

6.571.1.2 decaf::util::concurrent::MutexHandle::~~MutexHandle ( ) [inline]

6.571.1.3 decaf::util::concurrent::MutexHandle::MutexHandle ( ) [inline]

6.571.1.4 decaf::util::concurrent::MutexHandle::~~MutexHandle ( ) [inline]

### 6.571.2 Field Documentation

6.571.2.1 volatile long long decaf::util::concurrent::MutexHandle::lock\_count

6.571.2.2 volatile long long decaf::util::concurrent::MutexHandle::lock\_owner

6.571.2.3 CRITICAL\_SECTION decaf::util::concurrent::MutexHandle::mutex

6.571.2.4 pthread\_mutex\_t decaf::util::concurrent::MutexHandle::mutex

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/MutexHandle.h
- src/main/decaf/internal/util/concurrent/windows/MutexHandle.h

## 6.572 decaf::internal::util::concurrent::MutexImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/MutexImpl.h>
```

### Static Public Member Functions

- static **decaf::util::concurrent::MutexHandle \* create ()**  
*Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.*
- static void **destroy (decaf::util::concurrent::MutexHandle \*handle)**  
*Destroy a previously create Mutex instance.*
- static void **lock (decaf::util::concurrent::MutexHandle \*handle)**  
*Locks the Mutex.*
- static bool **trylock (decaf::util::concurrent::MutexHandle \*handle)**  
*Tries to lock the Mutex.*
- static void **unlock (decaf::util::concurrent::MutexHandle \*handle)**  
*Unlocks the Mutex allowing other Thread to then acquire the Lock on it.*

### 6.572.1 Member Function Documentation

6.572.1.1 static decaf::util::concurrent::MutexHandle\*  
decaf::internal::util::concurrent::MutexImpl::create ( ) [static]

Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.

#### Returns

handle to a newly created Mutex.

6.572.1.2 static void decaf::internal::util::concurrent::MutexImpl::destroy (  
decaf::util::concurrent::MutexHandle \* handle ) [static]

Destroy a previously create Mutex instance.

**Parameters**

<i>mutex</i>	The Mutex instance to be destroyed.
--------------	-------------------------------------

6.572.1.3 `static void decaf::internal::util::concurrent::MutexImpl::lock (`  
`decaf::util::concurrent::MutexHandle * handle ) [static]`

Locks the Mutex.

If the Mutex is already locked by another thread this method blocks until the Mutex becomes unlocked and this thread acquires the lock.

**Parameters**

<i>handle</i>	the handle to the Mutex to Lock.
---------------	----------------------------------

6.572.1.4 `static bool decaf::internal::util::concurrent::MutexImpl::trylock (`  
`decaf::util::concurrent::MutexHandle * handle ) [static]`

Tries to lock the Mutex.

If the Mutex is unlocked this Thread acquires the lock on the Mutex and this method returns true, if the Mutex is already locked then the lock is not acquired and this method returns false.

**Parameters**

<i>handle</i>	the handle to the Mutex to attempt to Lock.
---------------	---

**Returns**

true if the lock was acquired false otherwise.

6.572.1.5 `static void decaf::internal::util::concurrent::MutexImpl::unlock (`  
`decaf::util::concurrent::MutexHandle * handle ) [static]`

Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

**Parameters**

<i>handle</i>	the handle to the Mutex to attempt to Lock.
---------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/MutexImpl.h`

## 6.573 decaf::internal::net::Network Class Reference

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

```
#include <src/main/decaf/internal/net/Network.h>
```

### Public Member Functions

- virtual `~Network ()`
- `decaf::util::concurrent::Mutex * getRuntimeLock ()`  
*Gets a pointer to the **Network** (p. 2744) Runtime's Lock object, this can be used by **Network** (p. 2744) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2744) layer, etc.*
- void `addNetworkResource (decaf::internal::util::Resource *value)`  
*Adds a Resource to the **Network** (p. 2744) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2744) Runtime are destroyed.*
- template<typename T >  
void `addAsResource (T *value)`

### Static Public Member Functions

- static `Network * getNetworkRuntime ()`  
*Gets the one and only instance of the **Network** (p. 2744) class, if this is called before the **Network** (p. 2744) layer has been initialized or after it has been shutdown then an `IllegalStateException` is thrown.*
- static void `initializeNetworking ()`  
*Initialize the Networking layer.*
- static void `shutdownNetworking ()`  
*Shutdown the **Network** (p. 2744) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.*

### Protected Member Functions

- `Network ()`

#### 6.573.1 Detailed Description

Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.

#### Since

1.0

### 6.573.2 Constructor & Destructor Documentation

6.573.2.1 `decaf::internal::net::Network::Network ( )` `[protected]`

6.573.2.2 `virtual decaf::internal::net::Network::~~Network ( )` `[virtual]`

### 6.573.3 Member Function Documentation

6.573.3.1 `template<typename T> void decaf::internal::net::Network::addAsResource ( T * value )` `[inline]`

6.573.3.2 `void decaf::internal::net::Network::addNetworkResource ( decaf::internal::util::Resource * value )`

Adds a Resource to the **Network** (p. 2744) Runtime, this resource will be held by the runtime until the Library shutdown method is called at which time all the Resources held by the **Network** (p. 2744) Runtime are destroyed.

#### Parameters

<i>value</i>	The Resource to add to the <b>Network</b> (p. 2744) Runtime.
--------------	--

#### Exceptions

<i>NullPointerException</i>	if the Resource value passed is null.
-----------------------------	---------------------------------------

6.573.3.3 `static Network* decaf::internal::net::Network::getNetworkRuntime ( )` `[static]`

Gets the one and only instance of the **Network** (p. 2744) class, if this is called before the **Network** (p. 2744) layer has been initialized or after it has been shutdown then an `IllegalStateException` is thrown.

#### Returns

pointer to the **Network** (p. 2744) runtime for the Decaf library.

6.573.3.4 `decaf::util::concurrent::Mutex* decaf::internal::net::Network::getRuntimeLock ( )`

Gets a pointer to the **Network** (p. 2744) Runtime's Lock object, this can be used by **Network** (p. 2744) layer APIs to synchronize around certain actions such as adding a resource to the **Network** (p. 2744) layer, etc.

The pointer returned is owned by the **Network** (p. 2744) runtime and should not be deleted or copied by the caller.

#### Returns

a pointer to the **Network** (p. 2744) Runtime's single Lock instance.



6.573.3.5 static void decaf::internal::net::Network::initializeNetworking ( ) [static]

Initialize the Networking layer.

6.573.3.6 static void decaf::internal::net::Network::shutdownNetworking ( ) [static]

Shutdown the **Network** (p. 2744) layer and free any associated resources, classes in the Decaf library that use the networking layer will now fail if used after calling the shutdown method.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**Network.h**

## 6.574 activemq::commands::NetworkBridgeFilter Class Reference

```
#include <src/main/activemq/commands/NetworkBridgeFilter.h>
```

Inheritance diagram for activemq::commands::NetworkBridgeFilter:

### Public Member Functions

- **NetworkBridgeFilter** ( )
- virtual ~**NetworkBridgeFilter** ( )
- virtual unsigned char **getDataStructureType** ( ) const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **NetworkBridgeFilter** \* **cloneDataStructure** ( ) const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** ( ) const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual int **getNetworkTTL** ( ) const
- virtual void **setNetworkTTL** (int networkTTL)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** ( ) const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ( )
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &networkBrokerId)

### Static Public Attributes

- static const unsigned char **ID\_NETWORKBRIDGEFILTER** = 91

### Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

## 6.574.1 Constructor & Destructor Documentation

6.574.1.1 **activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter** ( )

6.574.1.2 **virtual activemq::commands::NetworkBridgeFilter::~~NetworkBridgeFilter** ( )  
[virtual]

## 6.574.2 Member Function Documentation

6.574.2.1 **virtual NetworkBridgeFilter\* activemq::commands::NetworkBridgeFilter::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.574.2.2 **virtual void activemq::commands::NetworkBridgeFilter::copyDataStructure** ( const **DataStructure \* src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.574.2.3 `virtual bool activemq::commands::NetworkBridgeFilter::equals ( const  
DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.574.2.4 `virtual unsigned char activemq::commands::NetworkBridgeFilter::getDataStructureType  
( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.574.2.5 `virtual const Pointer<BrokerId>& ac-  
tivemq::commands::NetworkBridgeFilter::getNetworkBrokerId ( ) const [virtual]`

6.574.2.6 `virtual Pointer<BrokerId>& ac-  
tivemq::commands::NetworkBridgeFilter::getNetworkBrokerId ( ) [virtual]`

6.574.2.7 `virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL ( ) const  
[virtual]`

6.574.2.8 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId ( const  
Pointer< BrokerId > & networkBrokerId ) [virtual]`

6.574.2.9 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL ( int  
networkTTL ) [virtual]`

6.574.2.10 `virtual std::string activemq::commands::NetworkBridgeFilter::toString ( ) const  
[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

**Returns**

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

**6.574.3 Field Documentation**

6.574.3.1 **const unsigned char activemq::commands::NetworkBridgeFilter::ID\_ - NETWORKBRIDGEFILTER = 91** [static]

6.574.3.2 **Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId** [protected]

6.574.3.3 **int activemq::commands::NetworkBridgeFilter::networkTTL** [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**NetworkBridgeFilter.h**

**6.575 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference**

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2749).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFi
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller:

**Public Member Functions**

- **NetworkBridgeFilterMarshaller ()**
- virtual **~NetworkBridgeFilterMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*

6.575

**activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

**Class Reference**

2759

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.575.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2749).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.575.2 Constructor & Destructor Documentation

6.575.2.1 **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [*inline*]

6.575.2.2 **virtual activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [*inline, virtual*]

### 6.575.3 Member Function Documentation

6.575.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::createObject**  
( ) const [*virtual*]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.575.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::getDataStructureType(
    ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.575.3.3  virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseMarshal(
    OpenWireFormat * wireFormat, commands::DataStructure *
    dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.575.3.4  virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseUnmarshal(
    OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
    decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.575

**activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

**Class Reference**

2761

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.575.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.575.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.575.3.7 virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**NetworkBridgeFilterMarshaller.h**

## 6.576 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2753).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**:

#### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )



6.576

**activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**

**Class Reference**

**2763**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.576.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2753).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.576.2 Constructor & Destructor Documentation

6.576.2.1 **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [inline]

6.576.2.2 **virtual activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [inline, virtual]

### 6.576.3 Member Function Documentation

6.576.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.576.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.576.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.576.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.576

**activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**

**Class Reference**

2765

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.576.3.5 virtual int activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.576.3.6 virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.576.3.7 virtual void activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**NetworkBridgeFilterMarshaller.h**

## 6.577 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2757).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**:

#### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

6.577

**activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

**Class Reference**

**2767**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.577.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2757).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.577.2 Constructor & Destructor Documentation

6.577.2.1 **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [**inline**]

6.577.2.2 **virtual activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [**inline**, **virtual**]

### 6.577.3 Member Function Documentation

6.577.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.577.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.577.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.577.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.577

**activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

**Class Reference**

2769

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.577.3.5 virtual int activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.577.3.6 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.577.3.7 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**NetworkBridgeFilterMarshaller.h**

## 6.578 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2761).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**:

#### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )



## 6.578

**activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

### Class Reference

2771

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.578.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2761).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.578.2 Constructor & Destructor Documentation

6.578.2.1 **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [**inline**]

6.578.2.2 **virtual activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [**inline**, **virtual**]

### 6.578.3 Member Function Documentation

6.578.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.578.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.578.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.578.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.578

**activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

**Class Reference**

**2773**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.578.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.578.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.578.3.7 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**NetworkBridgeFilterMarshaller.h**

## 6.579 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2765).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**:

#### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

6.579

**activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

**Class Reference**

**2775**

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.579.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2765).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.579.2 Constructor & Destructor Documentation

6.579.2.1 **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [**inline**]

6.579.2.2 **virtual activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [**inline**, **virtual**]

### 6.579.3 Member Function Documentation

6.579.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.579.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.579.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.579.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.579

**activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

**Class Reference**

2777

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.579.3.5 virtual int activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.579.3.6 virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.579.3.7 virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**NetworkBridgeFilterMarshaller.h**

## 6.580 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2769).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFi
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**:

#### Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )



6.580

**activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

**Class Reference**

2779

---

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.580.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2769).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.580.2 Constructor & Destructor Documentation

6.580.2.1 **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller**  
( ) [**inline**]

6.580.2.2 **virtual activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller**  
( ) [**inline**, **virtual**]

### 6.580.3 Member Function Documentation

6.580.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::createObject**  
( ) **const** [**virtual**]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.580.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.580.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.580.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

6.580

**activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

**Class Reference**

**2781**

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

6.580.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.580.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.580.3.7 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**NetworkBridgeFilterMarshaller.h**

## 6.581 decaf::net::NoRouteToHostException Class Reference

```
#include <src/main/decaf/net/NoRouteToHostException.h>
```

Inheritance diagram for decaf::net::NoRouteToHostException:

#### Public Member Functions

- **NoRouteToHostException** () throw ()  
*Default Constructor.*
- **NoRouteToHostException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoRouteToHostException** (const **NoRouteToHostException** &ex) throw ()  
*Copy Constructor.*
- **NoRouteToHostException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoRouteToHostException** (const std::exception \*cause) throw ()  
*Constructor.*

- **NoRouteToHostException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoRouteToHostException** \* **clone** () const  
*Clones this exception.*
- virtual ~**NoRouteToHostException** () throw ()

### 6.581.1 Constructor & Destructor Documentation

6.581.1.1 decaf::net::NoRouteToHostException::NoRouteToHostException ( ) throw ()  
[inline]

Default Constructor.

6.581.1.2 decaf::net::NoRouteToHostException::NoRouteToHostException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.581.1.3 decaf::net::NoRouteToHostException::NoRouteToHostException ( const NoRouteToHostException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.581.1.4 decaf::net::NoRouteToHostException::NoRouteToHostException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.

<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.581.1.5 `decaf::net::NoRouteToHostException::NoRouteToHostException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.581.1.6 `decaf::net::NoRouteToHostException::NoRouteToHostException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.581.1.7 `virtual decaf::net::NoRouteToHostException::~~NoRouteToHostException ( ) throw () [inline, virtual]`

## 6.581.2 Member Function Documentation

6.581.2.1 `virtual NoRouteToHostException* decaf::net::NoRouteToHostException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3467).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**NoRouteToHostException.h**

## 6.582 decaf::security::NoSuchAlgorithmException Class Reference

```
#include <src/main/decaf/security/NoSuchAlgorithmException.h>
```

Inheritance diagram for decaf::security::NoSuchAlgorithmException:

### Public Member Functions

- **NoSuchAlgorithmException** () throw ()  
*Default Constructor.*
- **NoSuchAlgorithmException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoSuchAlgorithmException** (const **NoSuchAlgorithmException** &ex) throw ()  
*Copy Constructor.*
- **NoSuchAlgorithmException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchAlgorithmException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **NoSuchAlgorithmException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchAlgorithmException** \* **clone** () const  
*Clones this exception.*
- virtual ~**NoSuchAlgorithmException** () throw ()

### 6.582.1 Constructor & Destructor Documentation

6.582.1.1 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( ) throw ()  
[inline]

Default Constructor.

6.582.1.2 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.582.1.3 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const NoSuchAlgorithmException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.582.1.4 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.582.1.5 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--



## 6.583 decaf::lang::exceptions::NoSuchElementException Class Reference 2787

6.582.1.6 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.582.1.7 `virtual decaf::security::NoSuchAlgorithmException::~~NoSuchAlgorithmException ( ) throw () [inline, virtual]`

## 6.582.2 Member Function Documentation

6.582.2.1 `virtual NoSuchAlgorithmException* decaf::security::NoSuchAlgorithmException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1936).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchAlgorithmException.h`

## 6.583 decaf::lang::exceptions::NoSuchElementException Class Reference

```
#include <src/main/decaf/lang/exceptions/NoSuchElementException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NoSuchElementException`:

## Public Member Functions

- **NoSuchElementException** () throw ()  
*Default Constructor.*
- **NoSuchElementException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **NoSuchElementException** (const **NoSuchElementException** &ex) throw ()  
*Copy Constructor.*
- **NoSuchElementException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchElementException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoSuchElementException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchElementException** \* **clone** () const  
*Clones this exception.*
- virtual ~**NoSuchElementException** () throw ()

## 6.583.1 Constructor & Destructor Documentation

6.583.1.1 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( ) throw () [inline]`

Default Constructor.

6.583.1.2 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1794).

### Parameters

<code>ex</code>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------------	--

6.583.1.3 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const NoSuchElementException & ex ) throw () [inline]`

Copy Constructor.

### Parameters

<code>ex</code>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------------	--

## 6.583 decaf::lang::exceptions::NoSuchElementException Class Reference 2789

6.583.1.4 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const char \* *file*, const int *lineNumber*, const std::exception \* *cause*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.583.1.5 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const std::exception \* *cause* ) throw () [inline]

Constructor.

### Parameters

<i>cause</i>	<b>Pointer</b> (p.2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.583.1.6 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.583.1.7 virtual decaf::lang::exceptions::NoSuchElementException::~~NoSuchElementException ( ) throw () [inline, virtual]

## 6.583.2 Member Function Documentation

```
6.583.2.1 virtual NoSuchElementException* de-
caf::lang::exceptions::NoSuchElementException::clone ( )
const [inline, virtual]
```

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NoSuchElementException.h**

## 6.584 decaf::security::NoSuchProviderException Class Reference

```
#include <src/main/decaf/security/NoSuchProviderException.h>
```

Inheritance diagram for decaf::security::NoSuchProviderException:

### Public Member Functions

- **NoSuchProviderException** () throw ()  
*Default Constructor.*
- **NoSuchProviderException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **NoSuchProviderException** (const **NoSuchProviderException** &ex) throw ()  
*Copy Constructor.*
- **NoSuchProviderException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NoSuchProviderException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NoSuchProviderException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NoSuchProviderException** \* **clone** () const  
*Clones this exception.*
- virtual ~**NoSuchProviderException** () throw ()

### 6.584.1 Constructor & Destructor Documentation

6.584.1.1 `decaf::security::NoSuchProviderException::NoSuchProviderException ( ) throw ()`  
`[inline]`

Default Constructor.

6.584.1.2 `decaf::security::NoSuchProviderException::NoSuchProviderException ( const Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.584.1.3 `decaf::security::NoSuchProviderException::NoSuchProviderException ( const NoSuchProviderException & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.584.1.4 `decaf::security::NoSuchProviderException::NoSuchProviderException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.584.1.5 `decaf::security::NoSuchProviderException::NoSuchProviderException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.584.1.6 `decaf::security::NoSuchProviderException::NoSuchProviderException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

6.584.1.7 `virtual decaf::security::NoSuchProviderException::~~NoSuchProviderException ( ) throw () [inline, virtual]`

**6.584.2 Member Function Documentation**

6.584.2.1 `virtual NoSuchProviderException* decaf::security::NoSuchProviderException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1936).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchProviderException.h`

**6.585 decaf::lang::exceptions::NullPointerException Class Reference**

```
#include <src/main/decaf/lang/exceptions/NullPointerException.h>
```

Inheritance diagram for decaf::lang::exceptions::NullPointerException:

## Public Member Functions

- **NullPointerException** () throw ()  
*Default Constructor.*
- **NullPointerException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()  
*Copy Constructor.*
- **NullPointerException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NullPointerException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NullPointerException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NullPointerException** \* **clone** () const  
*Clones this exception.*
- virtual ~**NullPointerException** () throw ()

## 6.585.1 Constructor & Destructor Documentation

6.585.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException ( ) throw ()  
[inline]

Default Constructor.

6.585.1.2 decaf::lang::exceptions::NullPointerException::NullPointerException ( const **Exception** & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1794).

### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.585.1.3 decaf::lang::exceptions::NullPointerException::NullPointerException ( const **NullPointerException** & ex ) throw () [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.585.1.4 `decaf::lang::exceptions::NullPointerException::NullPointerException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.585.1.5 `decaf::lang::exceptions::NullPointerException::NullPointerException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.585.1.6 `decaf::lang::exceptions::NullPointerException::NullPointerException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message



6.585.1.7 virtual decaf::lang::exceptions::NullPointerException::~~NullPointerException ( )  
throw () [inline, virtual]

## 6.585.2 Member Function Documentation

6.585.2.1 virtual NullPointerException\* decaf::lang::exceptions::NullPointerException::clone ( ) const  
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NullPointerException.h**

## 6.586 decaf::lang::Number Class Reference

The abstract class **Number** (p. 2786) is the superclass of classes **Byte** (p. 918), **Double** (p. 1751), **Float** (p. 1865), **Integer** (p. 2038), **Long** (p. 2377), and **Short** (p. 3380).

```
#include <src/main/decaf/lang/Number.h>
```

Inheritance diagram for decaf::lang::Number:

### Public Member Functions

- virtual ~**Number** ()
- virtual unsigned char **byteValue** () const  
*Answers the byte value which the receiver represents.*
- virtual double **doubleValue** () const =0  
*Answers the double value which the receiver represents.*
- virtual float **floatValue** () const =0  
*Answers the float value which the receiver represents.*
- virtual int **intValue** () const =0  
*Answers the int value which the receiver represents.*
- virtual long long **longValue** () const =0

*Answers the long value which the receiver represents.*

- virtual short **shortValue** () const

*Answers the short value which the receiver represents.*

### 6.586.1 Detailed Description

The abstract class **Number** (p. 2786) is the superclass of classes **Byte** (p. 918), **Double** (p. 1751), **Float** (p. 1865), **Integer** (p. 2038), **Long** (p. 2377), and **Short** (p. 3380).

Subclasses of **Number** (p. 2786) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

### 6.586.2 Constructor & Destructor Documentation

6.586.2.1 virtual decaf::lang::Number::~~Number ( ) [inline, virtual]

### 6.586.3 Member Function Documentation

6.586.3.1 virtual unsigned char decaf::lang::Number::byteValue ( ) const [inline, virtual]

Answers the byte value which the receiver represents.

#### Returns

byte the value of the receiver.

Reimplemented in **decaf::lang::Byte** (p. 921), **decaf::lang::Character** (p. 1071), **decaf::lang::Double** (p. 1753), **decaf::lang::Float** (p. 1867), **decaf::lang::Integer** (p. 2042), **decaf::lang::Long** (p. 2380), and **decaf::lang::Short** (p. 3382).

6.586.3.2 virtual double decaf::lang::Number::doubleValue ( ) const [pure virtual]

Answers the double value which the receiver represents.

#### Returns

double the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 922), **decaf::lang::Character** (p. 1072), **decaf::lang::Double** (p. 1755), **decaf::lang::Float** (p. 1868), **decaf::lang::Integer** (p. 2043), **decaf::lang::Long** (p. 2382), **decaf::lang::Short** (p. 3384), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 710).

6.586.3.3 virtual float decaf::lang::Number::floatValue ( ) const [pure virtual]

Answers the float value which the receiver represents.

### Returns

float the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 923), **decaf::lang::Character** (p. 1073), **decaf::lang::Double** (p. 1756), **decaf::lang::Float** (p. 1870), **decaf::lang::Integer** (p. 2044), **decaf::lang::Long** (p. 2382), **decaf::lang::Short** (p. 3384), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 711).

6.586.3.4 `virtual int decaf::lang::Number::intValue ( ) const [pure virtual]`

Answers the int value which the receiver represents.

### Returns

int the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 923), **decaf::lang::Character** (p. 1073), **decaf::lang::Double** (p. 1756), **decaf::lang::Float** (p. 1871), **decaf::lang::Integer** (p. 2044), **decaf::lang::Long** (p. 2383), **decaf::lang::Short** (p. 3385), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 712).

6.586.3.5 `virtual long long decaf::lang::Number::longValue ( ) const [pure virtual]`

Answers the long value which the receiver represents.

### Returns

long long the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 923), **decaf::lang::Character** (p. 1074), **decaf::lang::Double** (p. 1758), **decaf::lang::Float** (p. 1872), **decaf::lang::Integer** (p. 2044), **decaf::lang::Long** (p. 2383), **decaf::lang::Short** (p. 3385), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 712).

6.586.3.6 `virtual short decaf::lang::Number::shortValue ( ) const [inline, virtual]`

Answers the short value which the receiver represents.

### Returns

short the value of the receiver.

Reimplemented in **decaf::lang::Byte** (p. 926), **decaf::lang::Character** (p. 1076), **decaf::lang::Double** (p. 1759), **decaf::lang::Float** (p. 1874), **decaf::lang::Integer** (p. 2050), **decaf::lang::Long** (p. 2388), and **decaf::lang::Short** (p. 3388).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Number.h`

## 6.587 decaf::lang::exceptions::NumberFormatException Class Reference

```
#include <src/main/decaf/lang/exceptions/NumberFormatException.h>
```

Inheritance diagram for decaf::lang::exceptions::NumberFormatException:

### Public Member Functions

- **NumberFormatException** ()  
*Default Constructor.*
- **NumberFormatException** (const **Exception** &ex) throw ()  
*Conversion Constructor from some other **Exception** (p. 1794).*
- **NumberFormatException** (const **NumberFormatException** &ex) throw ()  
*Copy Constructor.*
- **NumberFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **NumberFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **NumberFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **NumberFormatException** \* clone () const  
*Clones this exception.*
- virtual ~**NumberFormatException** () throw ()

### 6.587.1 Constructor & Destructor Documentation

6.587.1.1 decaf::lang::exceptions::NumberFormatException::NumberFormatException ( )  
[inline]

Default Constructor.

Referenced by clone().

6.587.1.2 decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const **Exception** & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1794).

#### Parameters

ex	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
----	--

## 6.587 decaf::lang::exceptions::NumberFormatException Class Reference 2799

6.587.1.3 decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const NumberFormatException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.587.1.4 decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

6.587.1.5 decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const std::exception \* cause ) throw () [inline]

Constructor.

### Parameters

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.587.1.6 decaf::lang::exceptions::NumberFormatException::NumberFormatException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
-------------	--------------------------------------

<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

References `decaf::lang::Exception::buildMessage()`, and `decaf::lang::Exception::setMark()`.

6.587.1.7 `virtual decaf::lang::exceptions::NumberFormatException::~~NumberFormatException  
( ) throw () [inline, virtual]`

## 6.587.2 Member Function Documentation

6.587.2.1 `virtual NumberFormatException* de-  
caf::lang::exceptions::NumberFormatException::clone ( ) const  
[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

References `NumberFormatException()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NumberFormatException.h`

## 6.588 cms::ObjectMessage Class Reference

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

```
#include <src/main/cms/ObjectMessage.h>
```

Inheritance diagram for `cms::ObjectMessage`:

### Public Member Functions

- `virtual ~ObjectMessage ()`

6.588.1 Detailed Description

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

serialized **ObjectMessage** (p. 2791) s.

Since

1.0

6.588.2 Constructor & Destructor Documentation

6.588.2.1 virtual cms::ObjectMessage::~ObjectMessage ( ) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**ObjectMessage.h**

6.589 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference

Provides an SSLContext that wraps the OpenSSL API.

#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h>

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLContextSpi:

Public Member Functions

- **OpenSSLContextSpi** ()
- virtual ~**OpenSSLContextSpi** ()
- virtual void **providerInit** (**security::SecureRandom** \*random)

*Perform the initialization of this Context.*

**Parameters**

random	Pointer to an instance of a secure random number generator.
--------	---

**Exceptions**

NullPointerException	if the SecureRandom instance is NULL.
KeyManagementException	if an error occurs while initializing the context.

- virtual **decaf::net::SocketFactory** \* **providerGetSocketFactory** ()  
*Returns a **SocketFactory** (p. 3467) instance that can be used to create new **SSLSocket** (p. 3506) objects.*

The **SocketFactory** (p. 3467) is owned by the Service Provider and should not be destroyed by the caller.

#### Returns

**SocketFactory** (p. 3467) instance that can be used to create new SSLSockets.

#### Exceptions

IllegalStateException	if the <b>SSLContextSpi</b> (p. 3492) object requires initialization but has not been initialized yet.
-----------------------	--

- virtual **decaf::net::ServerSocketFactory \* providerGetServerSocketFactory**  
( )

Returns a **ServerSocketFactory** (p. 3301) instance that can be used to create new **SSLServerSocket** (p. 3498) objects.

The **ServerSocketFactory** (p. 3301) is owned by the Service Provider and should not be destroyed by the caller.

#### Returns

**SocketFactory** (p. 3467) instance that can be used to create new SSLServerSockets.

#### Exceptions

IllegalStateException	if the <b>SSLContextSpi</b> (p. 3492) object requires initialization but has not been initialized yet.
-----------------------	--

## Friends

- class **OpenSSLSocket**
- class **OpenSSLSocketFactory**

## 6.589.1 Detailed Description

Provides an SSLContext that wraps the OpenSSL API.

### Since

1.0

## 6.589.2 Constructor & Destructor Documentation

6.589.2.1 **decaf::internal::net::ssl::openssl::OpenSSLContextSpi::OpenSSLContextSpi** ( )

6.589.2.2 **virtual decaf::internal::net::ssl::openssl::OpenSSLContextSpi::~~OpenSSLContextSpi**  
( ) [virtual]

## 6.589.3 Member Function Documentation



## 6.589 decaf::internal::net::ssl::openssl::OpenSSLContextSpi Class Reference 2803

6.589.3.1 virtual decaf::net::ServerSocketFactory\* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetServerSocketFactory ( ) [virtual]

Returns a **ServerSocketFactory** (p. 3301) instance that can be used to create new **SSLServerSocket** (p. 3498) objects.

The **ServerSocketFactory** (p. 3301) is owned by the Service Provider and should not be destroyed by the caller.

### Returns

**SocketFactory** (p. 3467) instance that can be used to create new **SSLServerSockets**.

### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3492) object requires initialization but has not been initialized yet.
------------------------------	--

Implements **decaf::net::ssl::SSLContextSpi** (p. 3493).

6.589.3.2 virtual decaf::net::SocketFactory\* decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerGetSocketFactory ( ) [virtual]

Returns a **SocketFactory** (p. 3467) instance that can be used to create new **SSLSocket** (p. 3506) objects.

The **SocketFactory** (p. 3467) is owned by the Service Provider and should not be destroyed by the caller.

### Returns

**SocketFactory** (p. 3467) instance that can be used to create new **SSLSockets**.

### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3492) object requires initialization but has not been initialized yet.
------------------------------	--

Implements **decaf::net::ssl::SSLContextSpi** (p. 3494).

6.589.3.3 virtual void decaf::internal::net::ssl::openssl::OpenSSLContextSpi::providerInit ( security::SecureRandom \* random ) [virtual]

Perform the initialization of this Context.

### Parameters

<i>random</i>	Pointer to an instance of a secure random number generator.
---------------	---

**Exceptions**

<i>NullPointerException</i>	if the SecureRandom instance is NULL.
<i>KeyManagementException</i>	if an error occurs while initializing the context.

Implements **decaf::net::ssl::SSLContextSpi** (p. 3494).

**6.589.4 Friends And Related Function Documentation**

**6.589.4.1 friend class OpenSSLSocket** [friend]

**6.589.4.2 friend class OpenSSLSocketFactory** [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLContextSpi.h**

**6.590 decaf::internal::net::ssl::openssl::OpenSSLParameters Class Reference**

Container class for parameters that are Common to OpenSSL socket classes.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h>
```

**Public Member Functions**

- virtual **~OpenSSLParameters** ()
- bool **getNeedClientAuth** () const
- void **setNeedClientAuth** (bool value)
- bool **getWantClientAuth** () const
- void **setWantClientAuth** (bool value)
- bool **getUseClientMode** () const
- void **setUseClientMode** (bool value)
- std::vector< std::string > **getSupportedCipherSuites** () const
- std::vector< std::string > **getSupportedProtocols** () const
- std::vector< std::string > **getEnabledCipherSuites** () const
- void **setEnabledCipherSuites** (const std::vector< std::string > &suites)
- std::vector< std::string > **getEnabledProtocols** () const
- void **setEnabledProtocols** (const std::vector< std::string > &protocols)
- **OpenSSLParameters \* clone** () const

*Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL\_CTX as this object's.*

### 6.590.1 Detailed Description

Container class for parameters that are Common to OpenSSL socket classes.

**Since**

1.0

### 6.590.2 Constructor & Destructor Documentation

6.590.2.1 `virtual decaf::internal::net::ssl::openssl::OpenSSLParameters::~~OpenSSLParameters ( ) [virtual]`

### 6.590.3 Member Function Documentation

6.590.3.1 `OpenSSLParameters* decaf::internal::net::ssl::openssl::OpenSSLParameters::clone ( ) const`

Creates a clone of this object such that all settings are transferred to a new instance of an SSL object whose parent is the same SSL\_CTX as this object's.

6.590.3.2 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledCipherSuites ( ) const`

6.590.3.3 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getEnabledProtocols ( ) const`

6.590.3.4 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getNeedClientAuth ( ) const [inline]`

6.590.3.5 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedCipherSuites ( ) const`

6.590.3.6 `std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLParameters::getSupportedProtocols ( ) const`

6.590.3.7 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getUseClientMode ( ) const [inline]`

6.590.3.8 `bool decaf::internal::net::ssl::openssl::OpenSSLParameters::getWantClientAuth ( ) const [inline]`

6.590.3.9 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledCipherSuites ( const std::vector< std::string > & suites )`

6.590.3.10 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setEnabledProtocols ( const std::vector< std::string > & protocols )`

6.590.3.11 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setNeedClientAuth (bool value) [inline]`

6.590.3.12 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setUseClientMode (bool value) [inline]`

6.590.3.13 `void decaf::internal::net::ssl::openssl::OpenSSLParameters::setWantClientAuth (bool value) [inline]`

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h`

## 6.591 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference

SSLServerSocket based on OpenSSL library code.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLServerSocket`:

### Public Member Functions

- **OpenSSLServerSocket** (**OpenSSLParameters** \*parameters)
- virtual **~OpenSSLServerSocket** ()
- virtual `std::vector< std::string >` **getSupportedCipherSuites** () const  
*Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3498).  
Normally not all of these cipher suites will be enabled on the **Socket** (p. 3445).*  
**Returns**  
*a vector containing the names of all the supported cipher suites.*
- virtual `std::vector< std::string >` **getSupportedProtocols** () const  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3498) instance.*  
**Returns**  
*a vector containing the names of all the supported protocols.*
- virtual `std::vector< std::string >` **getEnabledCipherSuites** () const  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3498).*  
**Returns**  
*vector of the names of all enabled Cipher Suites.*
- virtual void **setEnabledCipherSuites** (const `std::vector< std::string >` &suites)

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3498) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

#### Parameters

suites	An Vector of names for all the Cipher Suites that are to be enabled.
--------	--

#### Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.
--------------------------	--

- virtual std::vector< std::string > **getEnabledProtocols** () const

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3498).

#### Returns

vector of the names of all enabled Protocols.

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3498) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

#### Parameters

protocols	An Vector of names for all the Protocols that are to be enabled.
-----------	--

#### Exceptions

IllegalArgumentException	if the vector is empty or one of the names is invalid.
--------------------------	--

- virtual bool **getWantClientAuth** () const

#### Returns

true if the **Socket** (p. 3445) request client Authentication.

- virtual void **setWantClientAuth** (bool value)

Sets whether or not this **Socket** (p. 3445) will request Client Authentication.

If set to true the **Socket** (p. 3445) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

#### Parameters

value	Whether the server socket should request client authentication.
-------	---

- virtual bool **getNeedClientAuth** () const

#### Returns

true if the **Socket** (p. 3445) requires client Authentication.

- virtual void **setNeedClientAuth** (bool value)

Sets whether or not this **Socket** (p. 3445) will require Client Authentication.

If set to true the **Socket** (p. 3445) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

#### Parameters

value	Whether the server socket should require client authentication.
-------	---

- virtual **decaf::net::Socket** \* **accept** () throw ( decaf::io::IOException )

*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3292), the caller blocks until a connection is made.*

*If the **SO\_TIMEOUT** option is set this method could throw a **SocketTimeoutException** (p. 3487) if the operation times out.*

#### Returns

*a new **Socket** (p. 3445) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.*

#### Exceptions

IOException	<i>if an I/O error occurs while binding the socket.</i>
<b>SocketException</b> (p. 3465)	<i>if an error occurs while blocking on the accept call.</i>
<b>SocketTimeoutException</b> (p. 3487)	<i>if the <b>SO_TIMEOUT</b> option was used and the accept timed out.</i>

### 6.591.1 Detailed Description

SSLServerSocket based on OpenSSL library code.

#### Since

1.0

### 6.591.2 Constructor & Destructor Documentation

6.591.2.1 **decaf::internal::net::ssl::openssl::OpenSSLServerSocket::OpenSSLServerSocket ( OpenSSLParameters \* parameters )**

6.591.2.2 virtual **decaf::internal::net::ssl::openssl::OpenSSLServerSocket::~~OpenSSLServerSocket ( )** [virtual]

### 6.591.3 Member Function Documentation

6.591.3.1 virtual **decaf::net::Socket**\* **decaf::internal::net::ssl::openssl::OpenSSLServerSocket::accept ( )** throw ( decaf::io::IOException ) [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3292), the caller blocks until a connection is made.

If the **SO\_TIMEOUT** option is set this method could throw a **SocketTimeoutException** (p. 3487) if the operation times out.

#### Returns

a new **Socket** (p. 3445) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

## 6.591 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference 2809

### Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
<b>SocketException</b> (p. 3465)	if an error occurs while blocking on the accept call.
<b>SocketTimeoutException</b> (p. 3487)	if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented from **decaf::net::ServerSocket** (p. 3296).

6.591.3.2 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledCipherSuites ( )  
const [virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3498).

### Returns

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3501).

6.591.3.3 virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getEnabledProtocols ( )  
const [virtual]

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3498).

### Returns

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3501).

6.591.3.4 virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getNeedClientAuth ( )const [virtual]

### Returns

true if the **Socket** (p. 3445) requires client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3502).

```
6.591.3.5  virtual std::vector<std::string> de-
           caf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedCipherSuites ( )
           const [virtual]
```

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3498).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3445).

#### Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3502).

```
6.591.3.6  virtual std::vector<std::string> de-
           caf::internal::net::ssl::openssl::OpenSSLServerSocket::getSupportedProtocols ( )
           const [virtual]
```

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3498) instance.

#### Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3502).

```
6.591.3.7  virtual bool decaf::internal::net::ssl::openssl::OpenSSLServerSocket::getWantClientAuth
           ( ) const [virtual]
```

#### Returns

true if the **Socket** (p. 3445) request client Authentication.

Implements **decaf::net::ssl::SSLServerSocket** (p. 3502).

```
6.591.3.8  virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledCipherSuites
           ( const std::vector< std::string > & suites ) [virtual]
```

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3498) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

#### Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--



## 6.591 decaf::internal::net::ssl::openssl::OpenSSLServerSocket Class Reference

### Exceptions

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implements **decaf::net::ssl::SSLServerSocket** (p. 3503).

6.591.3.9 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setEnabledProtocols  
( const std::vector< std::string > & protocols ) [virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3498) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

### Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

### Exceptions

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implements **decaf::net::ssl::SSLServerSocket** (p. 3503).

6.591.3.10 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setNeedClientAuth  
( bool value ) [virtual]

Sets whether or not this **Socket** (p. 3445) will require Client Authentication.

If set to true the **Socket** (p. 3445) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

### Parameters

<i>value</i>	Whether the server socket should require client authentication.
--------------	---

Implements **decaf::net::ssl::SSLServerSocket** (p. 3503).

6.591.3.11 virtual void decaf::internal::net::ssl::openssl::OpenSSLServerSocket::setWantClientAuth  
( bool value ) [virtual]

Sets whether or not this **Socket** (p. 3445) will request Client Authentication.

If set to true the **Socket** (p. 3445) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

**Parameters**

<i>value</i>	Whether the server socket should request client authentication.
--------------	---

Implements **decaf::net::ssl::SSLServerSocket** (p. 3504).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLServerSocket.h**

## 6.592 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory

### Class Reference

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory:

**Public Member Functions**

- **OpenSSLServerSocketFactory** (**OpenSSLContextSpi** \*parent)
- virtual ~**OpenSSLServerSocketFactory** ()
- virtual **decaf::net::ServerSocket** \* **createServerSocket** ()

Create a new **ServerSocket** (p. 3292) that is unbound.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket** \* **createServerSocket** (int port)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
------	--

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket** \* **createServerSocket** (int port, int backlog)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.  
The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will use the specified connection backlog setting.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual **decaf::net::ServerSocket \* createServerSocket** (int port, int backlog, const **decaf::net::InetAddress \*address**)

Create a new **ServerSocket** (p. 3292) that is bound to the given port.  
The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3292) will listen on all interfaces.

**Parameters**

port	The port to bind the <b>ServerSocket</b> (p. 3292) to.
backlog	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.
address	The address of the interface on the local machine to bind to.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

IOException	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
-------------	---

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.  
Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3506)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.  
Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites()** (p. 3505)

### 6.592.1 Detailed Description

SSLServerSocketFactory that creates Server Sockets that use OpenSSL.

#### Since

1.0

### 6.592.2 Constructor & Destructor Documentation

6.592.2.1 `decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::OpenSSLServerSocketFactory ( OpenSSLContextSpi * parent )`

6.592.2.2 `virtual decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::~~OpenSSLServerSocketFactory ( ) [virtual]`

### 6.592.3 Member Function Documentation

6.592.3.1 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket ( ) [virtual]`

Create a new **ServerSocket** (p. 3292) that is unbound.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

#### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

#### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3302).

6.592.3.2 `virtual decaf::net::ServerSocket* decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket ( int port ) [virtual]`

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
-------------	--

## Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

## Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3303).

6.592.3.3 virtual **decaf::net::ServerSocket\*** **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket** (  
int *port*, int *backlog*, const **decaf::net::InetAddress** \* *address* ) [virtual]

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3292) will listen on all interfaces.

## Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

## Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

## Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3303).

6.592.3.4 virtual **decaf::net::ServerSocket\*** **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::createServerSocket** (  
int *port*, int *backlog* ) [virtual]

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will use the specified connection backlog setting.

## Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implements **decaf::net::ServerSocketFactory** (p. 3304).

```
6.592.3.5  virtual std::vector<std::string> de-
           caf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getDefaultCipherSuites
           ( ) [virtual]
```

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3506)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3505).

```
6.592.3.6  virtual std::vector<std::string> de-
           caf::internal::net::ssl::openssl::OpenSSLServerSocketFactory::getSupportedCipherSuites
           ( ) [virtual]
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites()** (p. 3505)

Implements **decaf::net::ssl::SSLServerSocketFactory** (p. 3506).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLServerSocketFactory.h**

## 6.593 decaf::internal::net::ssl::openssl::OpenSSLSocket Class Reference

Wraps a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocket:

### Public Member Functions

- **OpenSSLSocket** (**OpenSSLParameters** \*parameters)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const **decaf::net::InetAddress** \*address, int port)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const **decaf::net::InetAddress** \*address, int port, const **decaf::net::InetAddress** \*localAddress, int localPort)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const std::string &host, int port)
- **OpenSSLSocket** (**OpenSSLParameters** \*parameters, const std::string &host, int port, const **decaf::net::InetAddress** \*localAddress, int localPort)
- virtual ~**OpenSSLSocket** ()
- virtual void **connect** (const std::string &host, int port, int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

*Connects to the specified destination, with a specified timeout value.*

*If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3487) is thrown. A timeout value of zero is treated as an infinite timeout.*

#### Parameters

host	<i>The host name or IP address of the remote host to connect to.</i>
port	<i>The port on the remote host to connect to.</i>
timeout	<i>The number of Milliseconds to wait before treating the connection as failed.</i>

#### Exceptions

IOException	<i>Thrown if a failure occurred in the connect.</i>
<b>SocketTimeoutException</b> (p. 3487)	<i>if the timeout for connection is exceeded.</i>
IllegalArgumentExcep- tion	<i>if the timeout value is negative or the endpoint is invalid.</i>

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the **Socket** (p. 3445).*

*Once closed a **Socket** (p. 3445) cannot be connected or otherwise operated upon, a new **Socket** (p. 3445) instance must be created.*

#### Exceptions

IOException	<i>if an I/O error occurs while closing the <b>Socket</b> (p. 3445).</i>
-------------	--

- virtual **decaf::io::InputStream \* getInputStream ()** throw ( decaf::io::IOException )

*Gets the InputStream for this socket if its connected.*

*The pointer returned is the property of the associated **Socket** (p. 3445) and should not be deleted by the caller.*

*When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broker the read calls will normally throw an exception to indicate the failure.*

*Closing the InputStream will also close the underlying **Socket** (p. 3445).*

#### **Returns**

*The InputStream for this socket.*

#### **Exceptions**

IOException	<i>if an error occurs during creation of the InputStream, also if the <b>Socket</b> (p. 3445) is not connected or the input has been shutdown previously.</i>
-------------	---

- virtual **decaf::io::OutputStream \* getOutputStream ()** throw ( decaf::io::IOException )

*Gets the OutputStream for this socket if it is connected.*

*The pointer returned is the property of the **Socket** (p. 3445) instance and should not be deleted by the caller.*

*Closing the returned **Socket** (p. 3445) will also close the underlying **Socket** (p. 3445).*

#### **Returns**

*the OutputStream for this socket.*

#### **Exceptions**

IOException	<i>if an error occurs during the creation of this OutputStream, or if the <b>Socket</b> (p. 3445) is closed or the output has been shutdown previously.</i>
-------------	---

- virtual void **shutdownInput ()** throw ( decaf::io::IOException )

*Shuts down the InputStream for this socket essentially marking it as EOF.*

*The stream returns EOF for any calls to read after this method has been called.*

#### **Exceptions**

IOException	<i>if an I/O error occurs while performing this operation.</i>
-------------	--

- virtual void **shutdownOutput ()** throw ( decaf::io::IOException )

*Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OuputStream::write will throw an IOException.*

#### **Exceptions**

IOException	<i>if an I/O error occurs while performing this operation.</i>
-------------	--

- virtual void **setOOBInline (bool value)** throw ( decaf::net::SocketException )

*Sets the value of the OOBINLINE for this socket, by default this option is disabled.*

*If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.*

#### **Returns**

*true if OOBINLINE is enabled, false otherwise.*

#### **Exceptions**

<b>SocketException</b> (p. 3465)	<i>if an error is encountered while performing this operation.</i>
-------------------------------------	--



- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )

*Sends on byte of urgent data to the **Socket** (p. 3445).*

**Parameters**

data	The value to write as urgent data, only the lower eight bits are sent.
------	--

**Exceptions**

IOException	if an I/O error occurs while performing this operation.
-------------	---

- virtual std::vector< std::string > **getSupportedCipherSuites** () const

*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3506).*

*Normally not all of these cipher suites will be enabled on the **Socket** (p. 3445).*

**Returns**

*a vector containing the names of all the supported cipher suites.*

- virtual std::vector< std::string > **getSupportedProtocols** () const

*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3506) instance.*

**Returns**

*a vector containing the names of all the supported protocols.*

- virtual std::vector< std::string > **getEnabledCipherSuites** () const

*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3445).*

**Returns**

*vector of the names of all enabled Cipher Suites.*

- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)

*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3445) connection. Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.*

**Parameters**

suites	An Vector of names for all the Cipher Suites that are to be enabled.
--------	--

**Exceptions**

IllegalArgumentException	if the vector is empty or one of the names is invalid.
--------------------------	--

- virtual std::vector< std::string > **getEnabledProtocols** () const

*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3445).*

**Returns**

*vector of the names of all enabled Protocols.*

- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)

*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3445) connection. Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.*

**Parameters**

protocols	An Vector of names for all the Protocols that are to be enabled.
-----------	--

**Exceptions**

IllegalArgumentException	if the vector is empty or one of the names is invalid.
--------------------------	--

- virtual void **startHandshake** ()

*Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.*

*When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an IOException to indicate an error.*

**Exceptions**

IOException	if an I/O error occurs while performing the Handshake
-------------	---

- virtual void **setUseClientMode** (bool value)

*Determines the mode that the socket uses when a handshake is initiated, client or server.*

*This method must be called prior to any handshake attempts on this **Socket** (p. 3445), once a handshake has been initiated this socket remains the set mode; client or server, for the life of this object.*

**Parameters**

value	The mode setting, true for client or false for server.
-------	--

**Exceptions**

IllegalArgumentException	if the handshake process has begun and mode is locked.
--------------------------	--

- virtual bool **getUseClientMode** () const

*Gets whether this **Socket** (p. 3445) is in Client or Server mode, true indicates that the mode is set to Client.*

**Returns**

*true if the **Socket** (p. 3445) is in Client mode, false otherwise.*

- virtual void **setNeedClientAuth** (bool value)

*Sets the **Socket** (p. 3445) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*

*This option only applies to sockets in the Server mode.*

*If the option is enabled and the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.*

**Parameters**

value	The value indicating if a client is required to authenticate itself or not.
-------	---

- virtual bool **getNeedClientAuth** () const

*Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.*

*This option is only useful when the socket is operating in server mode.*

**Returns**

*true if client authentication is required.*

- virtual void **setWantClientAuth** (bool value)

Sets the **Socket** (p. 3445) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the `setNeedClientAuth` method.

#### Parameters

value	The value indicating if a client is requested to authenticate itself or not.
-------	--

- virtual bool **getWantClientAuth** () const

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

#### Returns

true if client authentication is required.

- int **read** (unsigned char \*buffer, int size, int offset, int length)

Reads the requested data from the Socket and write it into the passed in buffer.

- void **write** (const unsigned char \*buffer, int size, int offset, int length)

Writes the specified data in the passed in buffer to the Socket.

- int **available** ()

Gets the number of bytes in the Socket buffer that can be read without blocking.

### 6.593.1 Detailed Description

Wraps a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.

#### Since

1.0

### 6.593.2 Constructor & Destructor Documentation

6.593.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* parameters )

6.593.2.2 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* parameters, const decaf::net::InetAddress \*   
 address, int port )

6.593.2.3 decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket (   
 OpenSSLParameters \* parameters, const decaf::net::InetAddress \*   
 address, int port, const decaf::net::InetAddress \* localAddress, int localPort )

6.593.2.4 `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket ( OpenSSLParameters * parameters, const std::string & host, int port )`

6.593.2.5 `decaf::internal::net::ssl::openssl::OpenSSLSocket::OpenSSLSocket ( OpenSSLParameters * parameters, const std::string & host, int port, const decaf::net::InetAddress * localAddress, int localPort )`

6.593.2.6 `virtual decaf::internal::net::ssl::openssl::OpenSSLSocket::~~OpenSSLSocket ( )`  
[virtual]

### 6.593.3 Member Function Documentation

6.593.3.1 `int decaf::internal::net::ssl::openssl::OpenSSLSocket::available ( )`

Gets the number of bytes in the Socket buffer that can be read without blocking.

#### Returns

the number of bytes that can be read from the Socket without blocking.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

6.593.3.2 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::close ( ) throw ( decaf::io::IOException )` [virtual]

Closes the **Socket** (p. 3445).

Once closed a **Socket** (p. 3445) cannot be connected or otherwise operated upon, a new **Socket** (p. 3445) instance must be created.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while closing the <b>Socket</b> (p. 3445).
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 3452).

6.593.3.3 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::connect ( const std::string & host, int port, int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )` [virtual]

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3487) is thrown. A timeout value of zero is treated as an infinite timeout.

**Parameters**

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.
<i>timeout</i>	The number of Milliseconds to wait before treating the connection as failed.

**Exceptions**

<i>IOException</i>	Thrown if a failure occurred in the connect.
<b>SocketTimeoutException</b> (p. 3487)	if the timeout for connection is exceeded.
<i>IllegalArgumentEx-ception</i>	if the timeout value is negative or the endpoint is invalid.

Reimplemented from **decaf::net::Socket** (p. 3452).

6.593.3.4 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledCipherSuites ( ) const`  
[virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this SSL **Socket** (p. 3445).

**Returns**

vector of the names of all enabled Cipher Suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3510).

6.593.3.5 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getEnabledProtocols ( ) const`  
[virtual]

Returns a vector containing the names of all the currently enabled Protocols for this SSL **Socket** (p. 3445).

**Returns**

vector of the names of all enabled Protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3510).

6.593.3.6 `virtual decaf::io::InputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getInputStream ( ) throw ( decaf::io::IOException )` [virtual]

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 3445) and should not be deleted by the caller.

When the returned `InputStream` is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the `InputStream` will also close the underlying **Socket** (p. 3445).

### Returns

The `InputStream` for this socket.

### Exceptions

<i>IOException</i>	if an error occurs during creation of the <code>InputStream</code> , also if the <b>Socket</b> (p. 3445) is not connected or the input has been shutdown previously.
--------------------	--

Reimplemented from **decaf::net::Socket** (p. 3453).

**6.593.3.7** `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getNeedClientAuth ( ) const [virtual]`

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

### Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3511).

**6.593.3.8** `virtual decaf::io::OutputStream* decaf::internal::net::ssl::openssl::OpenSSLSocket::getOutputStream ( ) throw ( decaf::io::IOException ) [virtual]`

Gets the `OutputStream` for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3445) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3445) will also close the underlying **Socket** (p. 3445).

### Returns

the `OutputStream` for this socket.

### Exceptions

<i>IOException</i>	if an error occurs during the creation of this OutputStream, or if the <b>Socket</b> (p. 3445) is closed or the output has been shutdown previously.
--------------------	--

Reimplemented from **decaf::net::Socket** (p. 3455).

6.593.3.9 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedCipherSuites ( ) const [virtual]`

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3506).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3445).

#### Returns

a vector containing the names of all the supported cipher suites.

Implements **decaf::net::ssl::SSLSocket** (p. 3511).

6.593.3.10 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocket::getSupportedProtocols ( ) const [virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3506) instance.

#### Returns

a vector containing the names of all the supported protocols.

Implements **decaf::net::ssl::SSLSocket** (p. 3511).

6.593.3.11 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getUseClientMode ( ) const [virtual]`

Gets whether this **Socket** (p. 3445) is in Client or Server mode, true indicates that the mode is set to Client.

#### Returns

true if the **Socket** (p. 3445) is in Client mode, false otherwise.

Implements **decaf::net::ssl::SSLSocket** (p. 3512).

6.593.3.12 `virtual bool decaf::internal::net::ssl::openssl::OpenSSLSocket::getWantClientAuth ( ) const [virtual]`

Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

### Returns

true if client authentication is required.

Implements **decaf::net::ssl::SSLSocket** (p. 3512).

6.593.3.13 `int decaf::internal::net::ssl::openssl::OpenSSLSocket::read ( unsigned char * buffer, int size, int offset, int length )`

Reads the requested data from the Socket and write it into the passed in buffer.

### Parameters

<i>buffer</i>	The buffer to read into
<i>size</i>	The size of the specified buffer
<i>offset</i>	The offset into the buffer where reading should start filling.
<i>length</i>	The number of bytes past offset to fill with data.

### Returns

the actual number of bytes read or -1 if at EOF.

### Exceptions

<i>IOException</i>	if an I/O error occurs during the read.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

6.593.3.14 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::sendUrgentData ( int data ) throw ( decaf::io::IOException ) [virtual]`

Sends one byte of urgent data to the **Socket** (p. 3445).

### Parameters

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---



Reimplemented from **decaf::net::Socket** (p. 3458).

6.593.3.15 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledCipherSuites ( const std::vector< std::string > & suites ) [virtual]

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 3445) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

#### Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if the vector is empty or one of the names is invalid.
---------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3512).

6.593.3.16 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setEnabledProtocols ( const std::vector< std::string > & protocols ) [virtual]

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3445) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

#### Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if the vector is empty or one of the names is invalid.
---------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3513).

6.593.3.17 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setNeedClientAuth ( bool value ) [virtual]

Sets the **Socket** (p. 3445) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

**Parameters**

<i>value</i>	The value indicating if a client is required to authenticate itself or not.
--------------	---

Implements **decaf::net::ssl::SSLSocket** (p. 3513).

6.593.3.18 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setOOBInline ( bool value ) throw ( decaf::net::SocketException ) [virtual]`

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

**Returns**

true if OOBINLINE is enabled, false otherwise.

**Exceptions**

<b>SocketException</b> (p. 3465)	if an error is encountered while performing this operation.
-------------------------------------	---

Reimplemented from **decaf::net::Socket** (p. 3459).

6.593.3.19 `virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setUseClientMode ( bool value ) [virtual]`

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3445), once a handshake has be initiated this socket remains the the set mode; client or server, for the life of this object.

**Parameters**

<i>value</i>	The mode setting, true for client or false for server.
--------------	--

**Exceptions**

<i>IllegalArgumentEx-ception</i>	if the handshake process has begun and mode is locked.
----------------------------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3514).

6.593.3.20 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::setWantClientAuth (   
 bool *value* ) [virtual]

Sets the **Socket** (p. 3445) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid then the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the setNeedClientAuth method.

#### Parameters

<i>value</i>	The value indicating if a client is requested to authenticate itself or not.
--------------	--

Implements **decaf::net::ssl::SSLSocket** (p. 3514).

6.593.3.21 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownInput ( )   
 throw ( decaf::io::IOException ) [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 3462).

6.593.3.22 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::shutdownOutput ( )   
 throw ( decaf::io::IOException ) [virtual]

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OutputStream::write will throw an IOException.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented from **decaf::net::Socket** (p. 3463).

6.593.3.23 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocket::startHandshake ( )   
 [virtual]

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an `IOException` to indicate an error.

### Exceptions

<i>IOException</i>	if an I/O error occurs while performing the Handshake
--------------------	---

Implements **decaf::net::ssl::SSLSocket** (p. 3515).

6.593.3.24 `void decaf::internal::net::ssl::openssl::OpenSSLSocket::write ( const unsigned char * buffer, int size, int offset, int length )`

Writes the specified data in the passed in buffer to the Socket.

### Parameters

<i>buffer</i>	The buffer to write to the socket.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset into the buffer where the data to write starts at.
<i>length</i>	The number of bytes past offset to write.

### Exceptions

<i>IOException</i>	if an I/O error occurs during the write.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h`

## 6.594 decaf::internal::net::ssl::openssl::OpenSSLSocketException Class Reference

Subclass of the standard `SocketException` that knows how to produce an error message from the OpenSSL error stack.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLSocketException`:

### Public Member Functions

- **OpenSSLSocketException** () throw ()

Creates a new **OpenSSLSocketException** (p. 2821) with default values.

- **OpenSSLSocketException** (const Exception &ex) throw ()

Conversion Constructor from some other Exception.

- **OpenSSLSocketException** (const **OpenSSLSocketException** &ex) throw ()

Copy Constructor.

- **OpenSSLSocketException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

Create a new **OpenSSLSocketException** (p. 2821) and initializes the file name and line number where this message occurred.

- **OpenSSLSocketException** (const std::exception \*cause) throw ()

Creates a new **OpenSSLSocketException** (p. 2821) with the passed exception set as the cause of this exception.

- **OpenSSLSocketException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

Create a new **OpenSSLSocketException** (p. 2821) and initializes the file name and line number where this message occurred.

- **OpenSSLSocketException** (const char \*file, const int lineNumber) throw ()

Create a new **OpenSSLSocketException** (p. 2821) and initializes the file name and line number where this message occurred.

- virtual **OpenSSLSocketException** \* clone () const

Clones this exception.

- virtual ~**OpenSSLSocketException** () throw ()

## Protected Member Functions

- std::string **getErrorString** () const

Gets and formats an error message string from the OpenSSL error stack.

### 6.594.1 Detailed Description

Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.

Since

1.0

### 6.594.2 Constructor & Destructor Documentation

#### 6.594.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( ) throw ()

Creates a new **OpenSSLSocketException** (p. 2821) with default values.

6.594.2.2 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const Exception & ex ) throw ()`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An Exception object that should become this type of Exception.
-----------	--

6.594.2.3 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const OpenSSLSocketException & ex ) throw ()`

Copy Constructor.

#### Parameters

<i>ex</i>	The <b>OpenSSLSocketException</b> (p. 2821) whose values should be copied to this instance.
-----------	---

6.594.2.4 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`

Create a new **OpenSSLSocketException** (p. 2821) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown (can be null).
<i>msg</i>	The error message to report.
<i>...</i>	The list of primitives that are formatted into the message.

6.594.2.5 `decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const std::exception * cause ) throw ()`

Creates a new **OpenSSLSocketException** (p. 2821) with the passed exception set as the cause of this exception.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

## 6.594 decaf::internal::net::ssl::openssl::OpenSSLSocketException Class

### Reference

2833

6.594.2.6 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw ()

Create a new **OpenSSLSocketException** (p. 2821) and initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

#### Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The error message to report.
...	The list of primitives that are formatted into the message

6.594.2.7 decaf::internal::net::ssl::openssl::OpenSSLSocketException::OpenSSLSocketException ( const char \* *file*, const int *lineNumber* ) throw ()

Create a new **OpenSSLSocketException** (p. 2821) and initializes the file name and line number where this message occurred.

Sets the message to report by getting the complete set of error messages from the OpenSSL error stack and concatenating them into one string.

#### Parameters

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.

6.594.2.8 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketException::~~OpenSSLSocketException ( ) throw () [virtual]

### 6.594.3 Member Function Documentation

6.594.3.1 virtual **OpenSSLSocketException\*** decaf::internal::net::ssl::openssl::OpenSSLSocketException::clone ( ) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override this method.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3467).

6.594.3.2 `std::string decaf::internal::net::ssl::openssl::OpenSSLSocketException::getErrorString ( ) const` [protected]

Gets and formats an error message string from the OpenSSL error stack.

#### Returns

a string containing the complete OpenSSL error string.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h`

## 6.595 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory Class Reference

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory`:

### Public Member Functions

- **OpenSSLSocketFactory** (**OpenSSLContextSpi** \*parent)
- virtual **~OpenSSLSocketFactory** ()
- virtual **decaf::net::Socket** \* **createSocket** () throw ( decaf::io::IOException )

*Creates an unconnected **Socket** (p. 3445) object.*

#### Returns

*a new **Socket** (p. 3445) object, caller must free this object when done.*

#### Exceptions

IOException	if the <b>Socket</b> (p. 3445) cannot be created.
-------------	---

- virtual **decaf::net::Socket** \* **createSocket** (const **decaf::net::InetAddress** \*host, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

#### Parameters

host	The host to connect the socket to.
port	The port on the remote host to connect to.

#### Returns

*a new **Socket** (p. 3445) object, caller must free this object when done.*



**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const **decaf::net::InetAddress** \*host, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

The **Socket** (p. 3445) will be bound to the specified local address and port.

**Parameters**

host	The host to connect the socket to.
port	The port on the remote host to connect to.
ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
localPort	The local port to bind the <b>Socket</b> (p. 3445) to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &hostname, int port) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

**Parameters**

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual **decaf::net::Socket \* createSocket** (const std::string &name, int port, const **decaf::net::InetAddress** \*ifAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

**Parameters**

host	The host name or IP address to connect the socket to.
port	The port on the remote host to connect to.

ifAddress	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
localPort	The local port to bind the <b>Socket</b> (p. 3445) to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

IOException	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

- virtual std::vector< std::string > **getDefaultCipherSuites** ()

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3517)

- virtual std::vector< std::string > **getSupportedCipherSuites** ()

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites()** (p. 3517)

- virtual **decaf::net::Socket** \* **createSocket** (**decaf::net::Socket** \*socket, std::string host, int port, bool autoClose)

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

**Parameters**

socket	The existing socket to layer over.
host	The server host the original <b>Socket</b> (p. 3445) is connected to.
port	The server port the original <b>Socket</b> (p. 3445) is connected to.
autoClose	Should the layered over <b>Socket</b> (p. 3445) be closed when the topmost socket is closed.

**Returns**

a new **Socket** (p. 3445) instance that wraps the given **Socket** (p. 3445).

#### Exceptions

IOException	if an I/O exception occurs while performing this operation.
<b>UnknownHostException</b> (p. 3841)	if the host is unknown.

### 6.595.1 Detailed Description

Client Socket Factory that creates SSL based client sockets using the OpenSSL library.

#### Since

1.0

### 6.595.2 Constructor & Destructor Documentation

6.595.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::OpenSSLSocketFactory (   
OpenSSLContextSpi \* parent )

6.595.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::~~OpenSSLSocketFactory  
( ) [virtual]

### 6.595.3 Member Function Documentation

6.595.3.1 virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket  
( ) throw ( decaf::io::IOException ) [virtual]

Creates an unconnected **Socket** (p. 3445) object.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

IOException	if the <b>Socket</b> (p. 3445) cannot be created.
-------------	---

Reimplemented from **decaf::net::SocketFactory** (p. 3468).

6.595.3.2 virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket  
( decaf::net::Socket \* socket, std::string host, int port, bool autoClose )  
[virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer des-

tion. This socket is configured using the socket options established for this factory.

#### Parameters

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original <b>Socket</b> (p. 3445) is connected to.
<i>port</i>	The server port the original <b>Socket</b> (p. 3445) is connected to.
<i>autoClose</i>	Should the layered over <b>Socket</b> (p. 3445) be closed when the topmost socket is closed.

#### Returns

a new **Socket** (p. 3445) instance that wraps the given **Socket** (p. 3445).

#### Exceptions

<i>IOException</i>	if an I/O exception occurs while performing this operation.
<b>UnknownHostException</b> (p. 3841)	if the host is unknown.

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3516).

```
6.595.3.3  virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket
( const decaf::net::InetAddress * host, int port ) throw ( de-
caf::io::IOException, decaf::net::UnknownHostException )
[virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3469).

6.595.3.4 virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket  
( const std::string & *hostname*, int *port* ) throw ( decaf::io::IOException,  
decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

#### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3470).

6.595.3.5 virtual decaf::net::Socket\* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket  
( const std::string & *name*, int *port*, const decaf::net::InetAddress  
\* *ifAddress*, int *localPort* ) throw ( decaf::io::IOException,  
decaf::net::UnknownHostException ) [virtual]

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

#### Parameters

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3469).

```
6.595.3.6  virtual decaf::net::Socket* decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::createSocket
( const decaf::net::InetAddress * host, int port, const de-
  ccaf::net::InetAddress * ifAddress, int localPort ) throw (
  decaf::io::IOException, decaf::net::UnknownHostException )
[virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

The **Socket** (p. 3445) will be bound to the specified local address and port.

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implements **decaf::net::SocketFactory** (p. 3470).

```
6.595.3.7  virtual std::vector<std::string> de-
  ccaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getDefaultCipherSuites ( )
[virtual]
```

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

#### Returns

an STL vector containing the list of cipher suites enabled by default.

#### See also

**getSupportedCipherSuites()** (p. 3517)

Implements **decaf::net::ssl::SSLSocketFactory** (p. 3517).

6.595.3.8 `virtual std::vector<std::string> decaf::internal::net::ssl::openssl::OpenSSLSocketFactory::getSupportedCipherSuites ( ) [virtual]`

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

### Returns

an STL vector containing the list of supported cipher suites.

### See also

`getDefaultCipherSuites()` (p. 3517)

Implements `decaf::net::ssl::SSLSocketFactory` (p. 3517).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h`

## 6.596 `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream` Class Reference

An output stream for reading data from an OpenSSL Socket instance.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h>
```

Inheritance diagram for `decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream`:

### Public Member Functions

- `OpenSSLSocketInputStream (OpenSSLSocket *socket)`
- `virtual ~OpenSSLSocketInputStream ()`
- `virtual int available () const throw ( decaf::io::IOException )`

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

#### Returns

*the number of bytes available on this input stream.*

**Exceptions**

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual void **close** () throw ( decaf::io::IOException )

*Close - does nothing.*

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Not supported.*

**Protected Member Functions**

- virtual int **doReadByte** () throw ( io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**6.596.1 Detailed Description**

An output stream for reading data from an OpenSSL Socket instance.

**Since**

1.0

**6.596.2 Constructor & Destructor Documentation**

6.596.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::OpenSSLSocketInputStream ( OpenSSLSocket \* socket )

6.596.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::~~OpenSSLSocketInputStream ( ) [virtual]

**6.596.3 Member Function Documentation**

6.596.3.1 virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::available ( ) const throw ( decaf::io::IOException ) [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns**

the number of bytes available on this input stream.



## Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2004).

6.596.3.2 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::close ( )  
throw ( decaf::io::IOException ) [virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::InputStream** (p. 2004).

6.596.3.3 virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadArrayBounded  
( unsigned char \* *buffer*, int *size*, int *offset*, int  
*length* ) throw ( decaf::io::IOException, de-  
caf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException ) [protected,  
virtual]

Reimplemented from **decaf::io::InputStream** (p. 2005).

6.596.3.4 virtual int decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::doReadByte  
( ) throw ( io::IOException ) [protected, virtual]

Implements **decaf::io::InputStream** (p. 2005).

6.596.3.5 virtual long long decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream::skip  
( long long *num* ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Not supported.

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p.2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

Returns

total bytes skipped

Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p.2010).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/**OpenSSLSocketInputStream.h**

6.597 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream  
Class Reference

OutputStream implementation used to write data to an **OpenSSLSocket** (p.2808) instance.

```
#include <src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h>
```

Inheritance diagram for decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream:

Public Member Functions

- **OpenSSLSocketOutputStream** (**OpenSSLSocket** \*socket)
- virtual ~**OpenSSLSocketOutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
------------------------------	-----------------------------------

*The default implementation of this method does nothing.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.597.1 Detailed Description

OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2808) instance.

#### Since

1.0

### 6.597.2 Constructor & Destructor Documentation

6.597.2.1 decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::OpenSSLSocketOutputStream ( OpenSSLSocket \* socket )

6.597.2.2 virtual decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::~~OpenSSLSocketOutputStream ( ) [virtual]

### 6.597.3 Member Function Documentation

6.597.3.1 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::close ( ) throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2858).

6.597.3.2 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteArrayBounded ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
[protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2859).

6.597.3.3 virtual void decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream::doWriteByte ( unsigned char *c* ) throw ( decaf::io::IOException ) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2859).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h

## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormat.h>
```

Inheritance diagram for activemq::wireformat::openwire::OpenWireFormat:

### Public Member Functions

- **OpenWireFormat** (const **decaf::util::Properties** &properties)  
*Constructs a new **OpenWireFormat** (p. 2837) object.*
- virtual ~**OpenWireFormat** ()
- virtual bool **hasNegotiator** () const  
*Returns true if this **WireFormat** (p. 3907) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport) throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.*
- void **addMarshaller** (**marshal::DataStreamMarshaller** \*marshaller)  
*Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.*
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out) throw ( decaf::io::IOException )

*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*

- virtual **Pointer< commands::Command > unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in) throw ( **decaf::io::IOException** )

*Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.*

- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** \*object, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Utility method for Tight Marshaling the given object to the boolean stream passed.*

- void **tightMarshalNestedObject2** (**commands::DataStructure** \*o, **decaf::io::DataOutputStream** \*ds, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.*

- **commands::DataStructure** \* **tightUnmarshalNestedObject** (**decaf::io::DataInputStream** \*dis, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.*

- **commands::DataStructure** \* **looseUnmarshalNestedObject** (**decaf::io::DataInputStream** \*dis) throw ( **decaf::io::IOException** )

*Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.*

- void **looseMarshalNestedObject** (**commands::DataStructure** \*o, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Utility method to loosely Marshal an object that is derived from the DataStructure interface.*

- void **renegotiateWireFormat** (const **commands::WireFormatInfo** &info) throw ( **decaf::lang::exceptions::IllegalStateException** )

*Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.*

- virtual void **setPreferredWireFormatInfo** (const **Pointer< commands::WireFormatInfo >** &info) throw ( **decaf::lang::exceptions::IllegalStateException** )

*Configures this object using the provided WireformatInfo object.*

- virtual const **Pointer< commands::WireFormatInfo >** & **getPreferredWireFormatInfo** () const

*Gets the Preferred WireFormatInfo object that this class holds.*

- bool **isStackTraceEnabled** () const

*Checks if the stackTraceEnabled flag is on.*

- void **setStackTraceEnabled** (bool stackTraceEnabled)

*Sets if the stackTraceEnabled flag is on.*

- bool **isTcpNoDelayEnabled** () const

*Checks if the tcpNoDelayEnabled flag is on.*

- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)

*Sets if the tcpNoDelayEnabled flag is on.*

- int **getVersion** () const

*Get the current Wireformat Version.*

- void **setVersion** (int version) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Set the current Wireformat Version.*
- virtual bool **inReceive** () const  
*Is there a Message being unmarshaled?*
- bool **isCacheEnabled** () const  
*Checks if the cacheEnabled flag is on.*
- void **setCacheEnabled** (bool cacheEnabled)  
*Sets if the cacheEnabled flag is on.*
- int **getCacheSize** () const  
*Returns the currently set Cache size.*
- void **setCacheSize** (int value)  
*Sets the current Cache size.*
- bool **isTightEncodingEnabled** () const  
*Checks if the tightEncodingEnabled flag is on.*
- void **setTightEncodingEnabled** (bool tightEncodingEnabled)  
*Sets if the tightEncodingEnabled flag is on.*
- bool **isSizePrefixDisabled** () const  
*Checks if the sizePrefixDisabled flag is on.*
- void **setSizePrefixDisabled** (bool sizePrefixDisabled)  
*Sets if the sizePrefixDisabled flag is on.*
- long long **getMaxInactivityDuration** () const  
*Gets the MaxInactivityDuration setting.*
- void **setMaxInactivityDuration** (long long value)  
*Sets the MaxInactivityDuration setting.*
- long long **getMaxInactivityDurationInitialDelay** () const  
*Gets the MaxInactivityDurationInitialDelay setting.*
- void **setMaxInactivityDurationInitialDelay** (long long value)  
*Sets the MaxInactivityDurationInitialDelay setting.*

### Protected Member Functions

- **commands::DataStructure \* doUnmarshal** (decaf::io::DataInputStream \*dis) throw ( decaf::io::IOException )  
*Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.*
- void **destroyMarshallers** ()  
*Cleans up all registered Marshallers and empties the dataMarshallers vector.*

### Static Protected Attributes

- static const unsigned char **NULL\_TYPE**
- static const int **DEFAULT\_VERSION** = 1

## 6.598.1 Constructor & Destructor Documentation

6.598.1.1 `activemq::wireformat::openwire::OpenWireFormat ( const decaf::util::Properties & properties )`

Constructs a new **OpenWireFormat** (p. 2837) object.

### Parameters

<i>properties</i>	- can contain optional config params.
-------------------	---------------------------------------

6.598.1.2 `virtual activemq::wireformat::openwire::OpenWireFormat::~OpenWireFormat ( )`  
[virtual]

## 6.598.2 Member Function Documentation

6.598.2.1 `void activemq::wireformat::openwire::OpenWireFormat::addMarshaller ( marshal::DataStreamMarshaller * marshaller )`

Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.

### Parameters

<i>marshaller</i>	- the Marshaler to add to the collection.
-------------------	---

6.598.2.2 `virtual Pointer<transport::Transport> activemq::wireformat::openwire::OpenWireFormat::createNegotiator ( const Pointer< transport::Transport > & transport ) throw ( decaf::lang::exceptions::UnsupportedOperationException )`  
[virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

### Returns

new instance of a **WireFormatNegotiator** (p. 3946).

Implements **activemq::wireformat::WireFormat** (p. 3908).

6.598.2.3 `void activemq::wireformat::openwire::OpenWireFormat::destroyMarshallers ( )`  
[protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector.

This should be called before a reconfiguration of the version marshallers, or on destruction of this object

6.598.2.4 **commands::DataStructure\*** **activemq::wireformat::openwire::OpenWireFormat::doUnmarshal** (  
**decaf::io::DataInputStream \* *dis*** ) **throw ( decaf::io::IOException )**  
 [protected]

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

This method can return null if the type of the object to unmarshal is NULL, empty data.

#### Parameters

<i>dis</i>	- DataInputStream to read from
------------	--------------------------------

#### Returns

new DataStructure\* that the caller owns

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal
--------------------	---

6.598.2.5 **int** **activemq::wireformat::openwire::OpenWireFormat::getCacheSize** ( ) **const**  
 [inline]

Returns the currently set Cache size.

#### Returns

the current value of the broker's cache size.

6.598.2.6 **long long** **activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration**  
 ( ) **const** [inline]

Gets the MaxInactivityDuration setting.

#### Returns

maximum inactivity duration value in milliseconds.

6.598.2.7 **long long** **activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay**  
 ( ) **const** [inline]

Gets the MaxInactivityDurationInitialDelay setting.

#### Returns

maximum inactivity duration initial delay value in milliseconds.



**6.598.2.8** `virtual const Pointer<commands::WireFormatInfo>&  
activemq::wireformat::openwire::OpenWireFormat::getPreferredWireFormatInfo ( )  
const [inline, virtual]`

Gets the Preferred WireFormatInfo object that this class holds.

**Returns**

pointer to a preferred WireFormatInfo object

**6.598.2.9** `int activemq::wireformat::openwire::OpenWireFormat::getVersion ( ) const  
[inline, virtual]`

Get the current Wireformat Version.

**Returns**

int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 3909).

**6.598.2.10** `virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator ( )  
const [inline, virtual]`

Returns true if this **WireFormat** (p. 3907) has a Negotiator that needs to wrap the Transport that uses it.

**Returns**

true if the **WireFormat** (p. 3907) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3909).

**6.598.2.11** `virtual bool activemq::wireformat::openwire::OpenWireFormat::inReceive ( ) const  
[inline, virtual]`

Is there a Message being unmarshaled?

**Returns**

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3909).

6.598.2.12 `bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled ( ) const`  
`[inline]`

Checks if the cacheEnabled flag is on.

#### Returns

true if the flag is on.

6.598.2.13 `bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled ( )`  
`const [inline]`

Checks if the sizePrefixDisabled flag is on.

#### Returns

true if the flag is on.

6.598.2.14 `bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled ( )`  
`const [inline]`

Checks if the stackTraceEnabled flag is on.

#### Returns

true if the flag is on.

6.598.2.15 `bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled ( )`  
`const [inline]`

Checks if the tcpNoDelayEnabled flag is on.

#### Returns

true if the flag is on.

6.598.2.16 `bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled ( )`  
`const [inline]`

Checks if the tightEncodingEnabled flag is on.

#### Returns

true if the flag is on.

## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference 2853

6.598.2.17 void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject ( commands::DataStructure \* *o*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException )

Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.

The marshaled data is written to the passed in DataOutputStream.

### Parameters

<i>o</i>	- DataStructure derived Object to Marshal
<i>dataOut</i>	- DataOutputStream to write the data to

### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.598.2.18 commands::DataStructure\* activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject ( decaf::io::DataInputStream \* *dis* ) throw ( decaf::io::IOException )

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.

Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

### Parameters

<i>dis</i>	- the DataInputStream to read the data from
------------	---

### Returns

a new DataStructure derived Object pointer

### Exceptions

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.598.2.19 virtual void activemq::wireformat::openwire::OpenWireFormat::marshal ( const Pointer< commands::Command > & *command*, const activemq::transport::Transport \* *transport*, decaf::io::DataOutputStream \* *out* ) throw ( decaf::io::IOException )  
[virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

**Parameters**

<i>command</i>	The Command to Marshal.
<i>transport</i>	The Transport instance that called this method.
<i>out</i>	The output stream to write the command to.

**Exceptions**

<i>IOException</i>	
--------------------	--

Implements **activemq::wireformat::WireFormat** (p. 3910).

6.598.2.20 `void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat ( const commands::WireFormatInfo & info ) throw ( decaf::lang::exceptions::IllegalStateException )`

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

**Parameters**

<i>info</i>	- The new Wireformat Info settings
-------------	------------------------------------

**Exceptions**

<i>IllegalStateException</i>	is wire format can't be negotiated.
------------------------------	-------------------------------------

6.598.2.21 `void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled ( bool cacheEnabled ) [inline]`

Sets if the cacheEnabled flag is on.

**Parameters**

<i>cacheEnabled</i>	- true to turn flag is on
---------------------	---------------------------

6.598.2.22 `void activemq::wireformat::openwire::OpenWireFormat::setCacheSize ( int value ) [inline]`

Sets the current Cache size.

**Parameters**

<i>value</i>	- the value to send as the broker's cache size.
--------------	---

## 6.598 activemq::wireformat::openwire::OpenWireFormat Class Reference 2855

6.598.2.23 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration ( long long value ) [inline]`

Sets the MaxInactivityDuration setting.

### Parameters

<i>value</i>	- the Max inactivity duration value in milliseconds.
--------------	--

6.598.2.24 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay ( long long value ) [inline]`

Sets the MaxInactivityDurationInitialDelay setting.

### Parameters

<i>value</i>	- the Max inactivity Initial Delay duration value in milliseconds.
--------------	--

6.598.2.25 `virtual void activemq::wireformat::openwire::OpenWireFormat::setPreferredWireFormatInfo ( const Pointer< commands::WireFormatInfo > & info ) throw ( decaf::lang::exceptions::IllegalStateException ) [virtual]`

Configures this object using the provided WireFormatInfo object.

### Parameters

<i>info</i>	- a WireFormatInfo object, takes ownership.
-------------	---

6.598.2.26 `void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled ( bool sizePrefixDisabled ) [inline]`

Sets if the sizePrefixDisabled flag is on.

### Parameters

<i>sizePrefixDisabled</i>	- true to turn flag is on
---------------------------	---------------------------

6.598.2.27 `void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled ( bool stackTraceEnabled ) [inline]`

Sets if the stackTraceEnabled flag is on.

### Parameters

<i>stack-TraceEnabled</i>	- true to turn flag is on
---------------------------	---------------------------

6.598.2.28 `void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled ( bool tcpNoDelayEnabled ) [inline]`

Sets if the tcpNoDelayEnabled flag is on.

#### Parameters

<i>tcpNoDelayEnabled</i>	- true to turn flag is on
--------------------------	---------------------------

6.598.2.29 `void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled ( bool tightEncodingEnabled ) [inline]`

Sets if the tightEncodingEnabled flag is on.

#### Parameters

<i>tightEncodingEnabled</i>	- true to turn flag is on
-----------------------------	---------------------------

6.598.2.30 `void activemq::wireformat::openwire::OpenWireFormat::setVersion ( int version ) throw ( decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Set the current Wireformat Version.

#### Parameters

<i>version</i>	- int that identifies the version
----------------	-----------------------------------

Implements **activemq::wireformat::WireFormat** (p. 3910).

6.598.2.31 `virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1 ( commands::DataStructure * object, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Utility method for Tight Marshaling the given object to the boolean stream passed.

**Parameters**

<i>object</i>	- The DataStructure to marshal
<i>bs</i>	- the BooleanStream to write to

**Returns**

size of the data returned.

6.598.2.32 `void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2 ( commands::DataStructure * o, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`

Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

Writes the data to the Data Output Stream provided.

**Parameters**

<i>o</i>	- DataStructure object
<i>ds</i>	- DataOutputStream for writing
<i>bs</i>	- BooleanStream

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

6.598.2.33 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject ( decaf::io::DataInputStream * dis, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.

The DataStructure instance that is returned is now the property of the caller.

**Parameters**

<i>dis</i>	- DataInputStream to read from
<i>bs</i>	- BooleanStream to read from

**Returns**

Newly allocated DataStructure Object

**Exceptions**

<i>IOException</i>	if an error occurs.
--------------------	---------------------

```
6.598.2.34 virtual Pointer<commands::Command>
activemq::wireformat::openwire::OpenWireFormat::unmarshal ( const
activemq::transport::Transport * transport, decaf::io::DataInputStream
* in ) throw ( decaf::io::IOException ) [virtual]
```

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.

Returns a Pointer to the newly un-marshaled Command.

#### Parameters

<i>transport</i>	- Pointer to the transport that is making this request.
<i>in</i>	- the input stream to read the command from.

#### Returns

the newly marshaled Command, caller owns the pointer

#### Exceptions

<i>IOException</i>
--------------------

Implements **activemq::wireformat::WireFormat** (p. 3910).

### 6.598.3 Field Documentation

```
6.598.3.1 const int activemq::wireformat::openwire::OpenWireFormat::DEFAULT_
VERSION = 1 [static, protected]
```

```
6.598.3.2 const unsigned char activemq::wireformat::openwire::OpenWireFormat::NULL_
TYPE [static, protected]
```

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormat.h`

### 6.599 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireFormatFactory`:



## 6.599 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

### Public Member Functions

- **OpenWireFormatFactory** ()

*Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.*

- virtual **~OpenWireFormatFactory** ()
- virtual **Pointer< wireformat::WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw ( **decaf::lang::exceptions::IllegalStateException** )

*Creates a new **WireFormat** (p. 3907) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.599.1 Constructor & Destructor Documentation

6.599.1.1 **activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory** (  
) [inline]

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

URL options ----- `wireFormat.stackTraceEnabled` `wireFormat.cacheEnabled`  
`wireFormat.tcpNoDelayEnabled` `wireFormat.tightEncodingEnabled` `wireFormat.sizePrefixDisabled`  
`wireFormat.maxInactivityDuration` `wireFormat.maxInactivityDurationInitialDelay`

6.599.1.2 **virtual activemq::wireformat::openwire::OpenWireFormatFactory::~~OpenWireFormatFactory**  
( ) [inline, virtual]

### 6.599.2 Member Function Documentation

6.599.2.1 **virtual Pointer<wireformat::WireFormat>**  
**activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat**  
( const **decaf::util::Properties** & *properties* ) throw ( **decaf::lang::exceptions::IllegalStateException** ) [virtual]

Creates a new **WireFormat** (p. 3907) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters

<i>properties</i>	- the Properties for this <b>WireFormat</b> (p. 3907)
-------------------	---

Implements **activemq::wireformat::WireFormatFactory** (p. 3912).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h`

## 6.600 activemq::wireformat::openwire::OpenWireFormatNegotiator

### Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h>
```

Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatNegotiator:

#### Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** \*wireFormat, const **Pointer**< **transport::Transport** > &next)  
*Constructor - Initializes this object around another Transport.*
- virtual ~**OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer**< **commands::Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)  
*This is called in the context of the nested transport's reading thread.*
- virtual void **onTransportException** (**transport::Transport** \*source, const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*

#### 6.600.1 Constructor & Destructor Documentation

- 6.600.1.1 **activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator** (**OpenWireFormat** \* wireFormat, const **Pointer**< **transport::Transport** > & next )

Constructor - Initializes this object around another Transport.

## 6.600 activemq::wireformat::openwire::OpenWireFormatNegotiator Class

### Reference

2861

#### Parameters

<i>wireFormat</i>	- The <b>WireFormat</b> (p. 3907) object we use to negotiate
<i>next</i>	- The next transport in the chain

6.600.1.2 virtual `activemq::wireformat::openwire::OpenWireFormatNegotiator::~~OpenWireFormatNegotiator ( )` [virtual]

### 6.600.2 Member Function Documentation

6.600.2.1 virtual void `activemq::wireformat::openwire::OpenWireFormatNegotiator::close ( )`  
throw ( `decaf::io::IOException` ) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions

<i>IOException</i>	if errors occur.
--------------------	------------------

Reimplemented from `activemq::transport::TransportFilter` (p. 3829).

6.600.2.2 virtual void `activemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand ( const Pointer< commands::Command > & command )` [virtual]

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

#### Parameters

<i>command</i>	the received from the nested transport.
----------------	---

Reimplemented from `activemq::transport::TransportFilter` (p. 3832).

6.600.2.3 virtual void `activemq::wireformat::openwire::OpenWireFormatNegotiator::oneway ( const Pointer< commands::Command > & command )` throw ( `decaf::io::IOException`, `decaf::lang::exceptions::UnsupportedOperationException` ) [virtual]

Sends a one-way command.

Does not wait for any response from the broker. First waits for the `WireFormatInfo` exchange to happen so that we know how to encode out-bound data.

**Parameters**

<i>command</i>	the command to be sent.
----------------	-------------------------

**Exceptions**

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

```
6.600.2.4  virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException
            ( transport::Transport * source, const decaf::lang::Exception & ex )
            [virtual]
```

Event handler for an exception from a command transport.

**Parameters**

<i>source</i>	The source of the exception
<i>ex</i>	The exception.

```
6.600.2.5  virtual Pointer<commands::Response>
            activemq::wireformat::openwire::OpenWireFormatNegotiator::request
            ( const Pointer< commands::Command >
              & command ) throw ( decaf::io::IOException,
              decaf::lang::exceptions::UnsupportedOperationException )
            [virtual]
```

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

**Parameters**

<i>command</i>	The request to send.
----------------	----------------------

**Returns**

the response from the server.

**Exceptions**

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3833).

```
6.600.2.6 virtual Pointer<commands::Response>
activemq::wireformat::openwire::OpenWireFormatNegotiator::request
( const Pointer< commands::Command > & command,
  unsigned int timeout ) throw ( decaf::io::IOException,
  decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

#### Parameters

<i>command</i>	The request to send.
<i>timeout</i>	The time to wait for the response.

#### Returns

the response from the server.

#### Exceptions

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3833).

```
6.600.2.7 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::start ( )
throw ( decaf::io::IOException ) [virtual]
```

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

#### Exceptions

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 3834).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatNegotiator.h**

## 6.601 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireResponseBuilder`:

## Public Member Functions

- **OpenWireResponseBuilder** ()
- virtual **~OpenWireResponseBuilder** ()
- virtual **Pointer< commands::Response > buildResponse** (const **Pointer< commands::Command > &command**)
 

*Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.*
- virtual void **buildIncomingCommands** (const **Pointer< commands::Command > &command**, **decaf::util::StlQueue< Pointer< commands::Command > > &queue**)
 

*When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

## 6.601.1 Constructor & Destructor Documentation

6.601.1.1 `activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder ( )` [`inline`]

6.601.1.2 `virtual activemq::wireformat::openwire::OpenWireResponseBuilder::~~OpenWireResponseBuilder ( )` [`inline`, `virtual`]

## 6.601.2 Member Function Documentation

6.601.2.1 `virtual void activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands ( const Pointer< commands::Command > &command, decaf::util::StlQueue< Pointer< commands::Command > > &queue )` [`virtual`]

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

### Parameters

<i>command</i>	- The Command being sent to the Broker.
<i>queue</i>	- Queue of Command sent back from the broker.

Implements `activemq::transport::mock::ResponseBuilder` (p. 3232).

6.601.2.2 virtual **Pointer**<**commands::Response**>  
**activemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse** ( const  
**Pointer**< **commands::Command** > & *command* ) [virtual]

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

#### Parameters

<i>command</i>	- The command to build a response for
----------------	---------------------------------------

#### Returns

A Response object pointer, or NULL if no response.

Implements **activemq::transport::mock::ResponseBuilder** (p. 3232).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireResponseBuilder.h**

## 6.602 decaf::io::OutputStream Class Reference

Base interface for any class that wants to represent an output stream of bytes.

```
#include <src/main/decaf/io/OutputStream.h>
```

Inheritance diagram for decaf::io::OutputStream:

#### Public Member Functions

- **OutputStream** ()
- virtual **~OutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

##### Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
------------------------------	-----------------------------------

- virtual void **flush** () throw ( decaf::io::IOException )  
*Flushes this stream by writing any buffered output to the underlying stream.*

##### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------

- virtual void **write** (unsigned char c) throw ( decaf::io::IOException )  
*Writes a single byte to the output stream.*

- virtual void **write** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes an array of bytes to the output stream.*
- virtual void **write** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.*
- virtual std::string **toString** () const  
*Output a String representation of this object.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value)=0 throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )



### 6.602.1 Detailed Description

Base interface for any class that wants to represent an output stream of bytes.

#### Since

1.0

### 6.602.2 Constructor & Destructor Documentation

6.602.2.1 `decaf::io::OutputStream::OutputStream ( )`

6.602.2.2 `virtual decaf::io::OutputStream::~~OutputStream ( )` `[virtual]`

### 6.602.3 Member Function Documentation

6.602.3.1 `virtual void decaf::io::OutputStream::close ( )` `throw ( decaf::io::IOException )`  
`[virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

Implements **decaf::io::Closeable** (p. 1121).

Reimplemented in **decaf::internal::io::StandardErrorOutputStream** (p. 3523), **decaf::internal::io::StandardOutputStream** (p. 3526), **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream** (p. 2836), **decaf::internal::net::tcp::TcpSocketOutputStream** (p. 3695), **decaf::io::FilterOutputStream** (p. 1863), and **decaf::util::zip::DeflaterOutputStream** (p. 1685).

6.602.3.2 `virtual void decaf::io::OutputStream::doWriteArray ( const unsigned char * buffer, int size )` `throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented in **decaf::io::BufferedOutputStream** (p. 901), and **decaf::io::FilterOutputStream** (p. 1863).

6.602.3.3 `virtual void decaf::io::OutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[protected, virtual]`

Reimplemented in `activemq::io::LoggingOutputStream` (p. 2360), `decaf::internal::io::StandardErrorOutputStream` (p. 3523), `decaf::internal::io::StandardOutputStream` (p. 3526), `decaf::internal::net::ssl::openssl::OpenSSLOutputStream` (p. 2836), `decaf::internal::net::tcp::TcpSocketOutputStream` (p. 3695), `decaf::io::BufferedOutputStream` (p. 901), `decaf::io::ByteArrayOutputStream` (p. 993), `decaf::io::DataOutputStream` (p. 1548), `decaf::io::FilterOutputStream` (p. 1863), `decaf::util::zip::CheckedOutputStream` (p. 1114), and `decaf::util::zip::DeflaterOutputStream` (p. 1685).

6.602.3.4 `virtual void decaf::io::OutputStream::doWriteByte ( unsigned char value ) throw ( decaf::io::IOException )`  
`[protected, pure virtual]`

Implemented in `activemq::io::LoggingOutputStream` (p. 2360), `decaf::internal::io::StandardErrorOutputStream` (p. 3523), `decaf::internal::io::StandardOutputStream` (p. 3526), `decaf::internal::net::ssl::openssl::OpenSSLOutputStream` (p. 2837), `decaf::internal::net::tcp::TcpSocketOutputStream` (p. 3696), `decaf::io::BufferedOutputStream` (p. 901), `decaf::io::ByteArrayOutputStream` (p. 994), `decaf::io::DataOutputStream` (p. 1549), `decaf::io::FilterOutputStream` (p. 1863), `decaf::util::zip::CheckedOutputStream` (p. 1114), and `decaf::util::zip::DeflaterOutputStream` (p. 1685).

6.602.3.5 `virtual void decaf::io::OutputStream::flush ( ) throw ( decaf::io::IOException )`  
`[virtual]`

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

Implements `decaf::io::Flushable` (p. 1900).

Reimplemented in `decaf::internal::io::StandardErrorOutputStream` (p. 3523), `decaf::internal::io::StandardOutputStream` (p. 3526), `decaf::io::BufferedOutputStream` (p. 901), and `decaf::io::FilterOutputStream` (p. 1864).

6.602.3.6 `virtual void decaf::io::OutputStream::lock ( ) throw ( decaf::lang::exceptions::RuntimeException )`  
`[inline, virtual]`

Locks the object.

### Exceptions

<b><i>RuntimeException</i></b>	if an error occurs while locking the object.
--------------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

6.602.3.7 `virtual void decaf::io::OutputStream::notify ( ) throw  
( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

6.602.3.8 `virtual void decaf::io::OutputStream::notifyAll ( ) throw  
( decaf::lang::exceptions::RuntimeException,  
decaf::lang::exceptions::IllegalMonitorStateException ) [inline,  
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

6.602.3.9 `virtual std::string decaf::io::OutputStream::toString ( ) const [virtual]`

Output a String representation of this object.

The default version of this method just prints the Class Name.

#### Returns

a string representation of the object.

Reimplemented in **decaf::io::ByteArrayOutputStream** (p. 994), and **decaf::io::FilterOutputStream** (p. 1864).

6.602.3.10 `virtual bool decaf::io::OutputStream::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

6.602.3.11 `virtual void decaf::io::OutputStream::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline, virtual]`

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

6.602.3.12 `virtual void decaf::io::OutputStream::wait ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

```
6.602.3.13 virtual void decaf::io::OutputStream::wait ( long long millisecs
) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

```
6.602.3.14 virtual void decaf::io::OutputStream::wait ( long long millisecs, int
nanos ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.

<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

6.602.3.15 **virtual void decaf::io::OutputStream::write ( unsigned char *c* ) throw ( decaf::io::IOException )** [virtual]

Writes a single byte to the output stream.

The default implementation of this method calls the pure virtual method `doWriteByte` which must be implemented by any subclass of the **OutputStream** (p. 2856).

#### Parameters

<i>c</i>	The byte to write to the sink.
----------	--------------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

6.602.3.16 **virtual void decaf::io::OutputStream::write ( const unsigned char \* *buffer*, int *size* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )** [virtual]

Writes an array of bytes to the output stream.

The entire contents of the given vector are written to the output stream.

The default implementation of this method simply calls the `doWriteArray` which writes the contents of the array using the `doWriteByte` method repeatedly. It is recommended that a subclass override `doWriteArray` to provide more performant means of writing the array.

#### Parameters

<i>buffer</i>	The vector of bytes to write.
<i>size</i>	The size of the buffer passed.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<i>NullPointerException</i>	thrown if buffer is Null.
<i>IndexOutOfBoundsException</i>	if size value is negative.

6.602.3.17 virtual void decaf::io::OutputStream::write ( const unsigned char \* *buffer*,  
int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::NullPointerException,  
decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Writes an array of bytes to the output stream in order starting at buffer[offset] and proceeding until the number of bytes specified by the length argument are written or an error occurs.

The default implementation of this method simply calls the doWriteArrayBounded method which writes the contents of the array using the doWriteByte method repeatedly. It is recommended that a subclass override doWriteArrayBounded to provide more performant means of writing the array.

#### Parameters

<i>buffer</i>	The array of bytes to write.
<i>size</i>	The size of the buffer array passed.
<i>offset</i>	The position to start writing in buffer.
<i>length</i>	The number of bytes from the buffer to be written.

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<i>NullPointerException</i>	thrown if buffer is Null.
<i>IndexOutOfBoundsException</i>	if the offset + length > size. or one of the parameters is negative.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**OutputStream.h**

## 6.603 decaf::io::OutputStreamWriter Class Reference

A class for turning a character stream into a byte stream.

```
#include <src/main/decaf/io/OutputStreamWriter.h>
```

Inheritance diagram for decaf::io::OutputStreamWriter:

#### Public Member Functions

- **OutputStreamWriter** (**OutputStream** \*stream, bool own=false)  
*Creates a new OutputStreamWriter (p. 2864).*
- virtual ~**OutputStreamWriter** ()
- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.*

- virtual void **flush** () throw ( decaf::io::IOException )

*Flushes this stream by writing any buffered output to the underlying stream.*

### Protected Member Functions

- virtual void **doWriteArrayBounded** (const char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ).*

- virtual void **checkClosed** () const throw ( decaf::io::IOException )

### 6.603.1 Detailed Description

A class for turning a character stream into a byte stream.

#### See also

**InputStreamReader** (p. 2013)

#### Since

1.0

### 6.603.2 Constructor & Destructor Documentation

- 6.603.2.1 decaf::io::OutputStreamWriter::OutputStreamWriter ( **OutputStream** \* *stream*, bool *own* = false )

Creates a new **OutputStreamWriter** (p. 2864).

#### Parameters

<i>stream</i>	The <b>OutputStream</b> (p. 2856) to wrap. (cannot be NULL).
<i>own</i>	Indicates whether this instance own the given <b>OutputStream</b> (p. 2856). If true then the <b>OutputStream</b> (p. 2856) is destroyed when this class is.

#### Exceptions

<i>NullPointerException</i>	if the stream is NULL.
-----------------------------	------------------------

- 6.603.2.2 virtual decaf::io::OutputStreamWriter::~~OutputStreamWriter ( ) [virtual]

### 6.603.3 Member Function Documentation



6.603.3.1 virtual void decaf::io::OutputStreamWriter::checkClosed ( ) const throw ( decaf::io::IOException ) [protected, virtual]

6.603.3.2 virtual void decaf::io::OutputStreamWriter::close ( ) throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
---------------------------------	-----------------------------------

Implements **decaf::io::Closeable** (p. 1121).

6.603.3.3 virtual void decaf::io::OutputStreamWriter::doWriteArrayBounded ( const char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]

Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ).

All subclasses must override this method to provide the basic **Writer** (p. 3951) functionality.

Implements **decaf::io::Writer** (p. 3954).

6.603.3.4 virtual void decaf::io::OutputStreamWriter::flush ( ) throw ( decaf::io::IOException ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Implements **decaf::io::Flushable** (p. 1900).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**OutputStreamWriter.h**

## 6.604 activemq::commands::PartialCommand Class Reference

```
#include <src/main/activemq/commands/PartialCommand.h>
```

Inheritance diagram for activemq::commands::PartialCommand:

### Public Member Functions

- **PartialCommand** ()
- virtual **~PartialCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **PartialCommand** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual int **getCommandId** () const
- virtual void **setCommandId** (int commandId)
- virtual const std::vector< unsigned char > & **getData** () const
- virtual std::vector< unsigned char > & **getData** ()
- virtual void **setData** (const std::vector< unsigned char > &data)

### Static Public Attributes

- static const unsigned char **ID\_PARTIALCOMMAND** = 60

### Protected Attributes

- int **commandId**
- std::vector< unsigned char > **data**

### 6.604.1 Constructor & Destructor Documentation

6.604.1.1 `activemq::commands::PartialCommand::PartialCommand ( )`

6.604.1.2 `virtual activemq::commands::PartialCommand::~~PartialCommand ( )`  
[virtual]

### 6.604.2 Member Function Documentation

6.604.2.1 `virtual PartialCommand* activemq::commands::PartialCommand::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2261).

6.604.2.2 `virtual void activemq::commands::PartialCommand::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2261).

6.604.2.3 `virtual bool activemq::commands::PartialCommand::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2261).

6.604.2.4 `virtual int activemq::commands::PartialCommand::getCommandId ( ) const`  
[virtual]

6.604.2.5 `virtual std::vector<unsigned char>& activemq::commands::PartialCommand::getData ( )`  
[virtual]

6.604.2.6 `virtual const std::vector<unsigned char>& activemq::commands::PartialCommand::getData ( ) const`  
[virtual]

6.604.2.7 `virtual unsigned char activemq::commands::PartialCommand::getDataStructureType ( ) const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2262).

6.604.2.8 `virtual void activemq::commands::PartialCommand::setCommandId ( int commandId )` [virtual]

6.604.2.9 `virtual void activemq::commands::PartialCommand::setData ( const std::vector< unsigned char > & data )` [virtual]

6.604.2.10 `virtual std::string activemq::commands::PartialCommand::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 2262).

## 6.604.3 Field Documentation

## 6.605 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller

---

### Class Reference 2879

- 6.604.3.1 `int activemq::commands::PartialCommand::commandId`  
[protected]
- 6.604.3.2 `std::vector<unsigned char> activemq::commands::PartialCommand::data`  
[protected]
- 6.604.3.3 `const unsigned char activemq::commands::PartialCommand::ID_ - PARTIALCOMMAND = 60` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/PartialCommand.h`

## 6.605 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2870).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller`:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual `~PartialCommandMarshaller` ()
- virtual `commands::DataStructure * createObject` () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType` () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void `tightUnmarshal` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Un-marshal an object instance from the data input stream.*
- virtual int `tightMarshal1` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void `tightMarshal2` (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw ( `decaf::io::IOException` )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.605.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2870).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.605.2 Constructor & Destructor Documentation

- 6.605.2.1 **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::PartialCommandMarshaller**  
 ( ) [inline]
- 6.605.2.2 **virtual activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::~~PartialCommandMarshaller**  
 ( ) [inline, virtual]

### 6.605.3 Member Function Documentation

- 6.605.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2264).

- 6.605.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.605 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller

### Class Reference 2881

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2264).

```
6.605.3.3 virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2264).

```
6.605.3.4 virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2265).

```
6.605.3.5  virtual int activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2265).

```
6.605.3.6  virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2266).



## 6.606 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller

### Class Reference

2883

```
6.605.3.7 virtual void activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller** (p. 2266).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**PartialCommandMarshaller.h**

## 6.606 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2874).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**:

#### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual ~**PartialCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.606.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p.2874).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.606.2 Constructor & Destructor Documentation

- 6.606.2.1 **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::PartialCommandMarshaller**  
 ( ) [inline]
- 6.606.2.2 **virtual activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::~~PartialCommandMarshaller**  
 ( ) [inline, virtual]

### 6.606.3 Member Function Documentation

- 6.606.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

## 6.606 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller

### Class Reference 2885

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2272).

6.606.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2272).

6.606.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2272).

6.606.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2273).

```
6.606.3.5  virtual int activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2273).

```
6.606.3.6  virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

## 6.607 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller

### Class Reference

2887

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2274).

```
6.606.3.7 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 2274).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**PartialCommandMarshaller.h**

## 6.607 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2878).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandM
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller`:

## Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.607.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2878).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

### 6.607.2 Constructor & Destructor Documentation

6.607.2.1 `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::PartialCommandMarshaller`  
( ) [inline]

6.607.2.2 `virtual activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::~~PartialCommandMarshaller ( ) [inline, virtual]`

### 6.607.3 Member Function Documentation

6.607.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2276).

6.607.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2276).

6.607.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2276).

```
6.607.3.4  virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2277).

```
6.607.3.5  virtual int activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2277).

6.607.3.6 virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal2  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*,  
**decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw  
 ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2278).

6.607.3.7 virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightUnmarshal  
 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \*  
*dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \*  
*bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 2278).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h`

## 6.608 **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2883).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**:

### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.608.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2883).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.608.2 Constructor & Destructor Documentation

6.608.2.1 `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::PartialCommandMarshaller ( ) [inline]`

6.608.2.2 `virtual activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::~~PartialCommandMarshaller ( ) [inline, virtual]`

### 6.608.3 Member Function Documentation

6.608.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2268).

6.608.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2268).

```

6.608.3.3  virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2268).

```

6.608.3.4  virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2269).

## 6.608 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller

### Class Reference 2895

6.608.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2269).

6.608.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2270).

```
6.608.3.7 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 2270).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**PartialCommandMarshaller.h**

## 6.609 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2887).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandM
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**:

#### Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.609.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2887).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.609.2 Constructor & Destructor Documentation

- 6.609.2.1 **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::PartialCommandMarshaller**  
( ) [inline]
- 6.609.2.2 **virtual activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::~~PartialCommandMarshaller**  
( ) [inline, virtual]

### 6.609.3 Member Function Documentation

- 6.609.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2280).

6.609.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2280).

6.609.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2280).

6.609.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.



## 6.609 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller

### Class Reference 2899

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2281).

```
6.609.3.5 virtual int activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2281).

```
6.609.3.6 virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2282).

```
6.609.3.7 virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 2282).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**PartialCommandMarshaller.h**

## 6.610 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2891).

#include <src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller:

## Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.610.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2891).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.610.2 Constructor & Destructor Documentation

**6.610.2.1** **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::PartialCommandMarshaller**  
**( )** [inline]

6.610.2.2 `virtual activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::~~PartialCommandMarshaller ( ) [inline, virtual]`

### 6.610.3 Member Function Documentation

6.610.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2284).

6.610.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2284).

6.610.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2284).

6.610.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2285).

6.610.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2285).

```
6.610.3.6  virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 2286).

```
6.610.3.7  virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.611 `decaf::lang::Pointer< T, REFCOUNTER >` Class Template Reference 2905

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1620).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 2286).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h`

## 6.611 `decaf::lang::Pointer< T, REFCOUNTER >` Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/Pointer.h>
```

### Public Types

- `typedef T * PointerType`
- `typedef T & ReferenceType`
- `typedef REFCOUNTER CounterType`

### Public Member Functions

- **Pointer** ()  
*Default Constructor.*
- **Pointer** (const **PointerType** value)  
*Explicit Constructor, creates a **Pointer** (p. 2896) that contains value with a single reference.*
- **Pointer** (const **Pointer** &value) throw ()  
*Copy constructor.*
- `template<typename T1 , typename R1 >`  
**Pointer** (const **Pointer**< T1, R1 > &value) throw ()  
*Copy constructor.*
- `template<typename T1 , typename R1 >`  
**Pointer** (const **Pointer**< T1, R1 > &value, const **STATIC\_CAST\_TOKEN** &) throw ()  
*Static Cast constructor.*
- `template<typename T1 , typename R1 >`  
**Pointer** (const **Pointer**< T1, R1 > &value, const **DYNAMIC\_CAST\_TOKEN** &) throw ( `decaf::lang::exceptions::ClassCastException` )  
*Dynamic Cast constructor.*
- `virtual ~Pointer ()` throw ()
- `void reset (T *value)`

Resets the **Pointer** (p. 2896) to hold the new value.

- **T \* release** ()

Releases the **Pointer** (p. 2896) held and resets the internal pointer value to Null.

- **PointerType get** () const

Gets the real pointer that is contained within this **Pointer** (p. 2896).

- **void swap** (**Pointer** &value) throw ()

**Exception** (p. 1794) Safe Swap Function.

- **Pointer & operator=** (const **Pointer** &right) throw ()

Assigns the value of right to this **Pointer** (p. 2896) and increments the reference Count.

- **template**<typename T1 , typename R1 >

**Pointer & operator=** (const **Pointer**< T1, R1 > &right) throw ()

- **ReferenceType operator\*** ()

Dereference Operator, returns a reference to the Contained value.

- **ReferenceType operator\*** () const

- **PointerType operator->** ()

Indirection Operator, returns a pointer to the Contained value.

- **PointerType operator->** () const

- **bool operator!** () const

- **template**<typename T1 , typename R1 >

**bool operator==** (const **Pointer**< T1, R1 > &right) const

- **template**<typename T1 , typename R1 >

**bool operator!=** (const **Pointer**< T1, R1 > &right) const

- **template**<typename T1 >

**Pointer**< T1, **CounterType** > **dynamicCast** () const

- **template**<typename T1 >

**Pointer**< T1, **CounterType** > **staticCast** () const

## Friends

- **bool operator==** (const **Pointer** &left, const T \*right)

- **bool operator==** (const T \*left, const **Pointer** &right)

- **bool operator!=** (const **Pointer** &left, const T \*right)

- **bool operator!=** (const T \*left, const **Pointer** &right)

## 6.611.1 Detailed Description

```
template<typename T, typename REFCOUNTER = decaf::util::concurrent::atomic::AtomicRefCount> class
decaf::lang::Pointer< T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.

This **Pointer** (p. 2896) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of `RefCountCounter`.



## 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 2907

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2896) instances in the same manner as normal pointer, except that it does not provide an overload of operators ( <, <=, >, >= ). To allow use of a **Pointer** (p. 2896) in a STL container that requires it, **Pointer** (p. 2896) provides an implementation of std::less.

Since

1.0

### 6.611.2 Member Typedef Documentation

6.611.2.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef REFCOUNTER  
decaf::lang::Pointer< T, REFCOUNTER >::CounterType`

6.611.2.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef T*  
decaf::lang::Pointer< T, REFCOUNTER >::PointerType`

6.611.2.3 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> typedef T*  
decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType`

### 6.611.3 Constructor & Destructor Documentation

6.611.3.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer< T,  
REFCOUNTER >::Pointer ( ) [inline]`

Default Constructor.

Initialized the contained pointer to NULL, using the -> operator results in an exception unless reset to contain a real value.

Referenced by decaf::lang::Pointer< TransactionId >::reset().

6.611.3.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer<  
T, REFCOUNTER >::Pointer ( const PointerType value ) [inline,  
explicit]`

Explicit Constructor, creates a **Pointer** (p. 2896) that contains value with a single reference.

This object now has ownership until a call to release.

#### Parameters

<i>value</i>	- instance of the type we are containing here.
--------------	--

```
6.611.3.3  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> decaf::lang::Pointer< T,
            REFCOUNTER >::Pointer ( const Pointer< T, REFCOUNTER > & value ) throw ()
            [inline]
```

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

```
6.611.3.4  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,
            typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer ( const
            Pointer< T1, R1 > & value ) throw () [inline]
```

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

```
6.611.3.5  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,
            typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer ( const
            Pointer< T1, R1 > & value, const STATIC_CAST_TOKEN & ) throw ()
            [inline]
```

Static Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p. 2896) object.

#### Parameters

<i>value</i>	- <b>Pointer</b> (p. 2896) instance to cast to this type.
--------------	---

```
6.611.3.6  template<typename T, typename REFCOUNTER =
            decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,
            typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer ( const
            Pointer< T1, R1 > & value, const DYNAMIC_CAST_TOKEN & ) throw (
            decaf::lang::exceptions::ClassCastException ) [inline]
```

Dynamic Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p. 2896) object. If the cast fails and return NULL then this method throws a **ClassCastException**.

## 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 2909

### Parameters

<i>value</i>	- <b>Pointer</b> (p.2896) instance to cast to this type.
--------------	--

### Exceptions

<i>ClassCastException</i>	if the dynamic cast returns NULL
---------------------------	----------------------------------

6.611.3.7 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> virtual decaf::lang::Pointer<  
T, REFCOUNTER >::~~Pointer ( ) throw () [inline, virtual]`

### 6.611.4 Member Function Documentation

6.611.4.1 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 >  
Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER  
>::dynamicCast ( ) const [inline]`

6.611.4.2 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> PointerType  
decaf::lang::Pointer< T, REFCOUNTER >::get ( ) const [inline]`

Gets the real pointer that is contained within this **Pointer** (p.2896).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p.2896). Use at your own risk.

### Returns

the contained pointer.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::state::ConnectionState::getTransactionState()`, `decaf::lang::operator!=()`, `decaf::lang::Pointer< TransactionId >::operator!=()`, `std::less< decaf::lang::Pointer< T > >::operator()`, `decaf::lang::operator==()`, `decaf::lang::Pointer< TransactionId >::operator==()`, and `activemq::state::ConnectionState::removeTempDestination()`.

6.611.4.3 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> bool decaf::lang::Pointer< T,  
REFCOUNTER >::operator! ( ) const [inline]`

6.611.4.4 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCounter> template<typename T1 ,  
typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator!= (   
const Pointer< T1, R1 > & right ) const [inline]`

```
6.611.4.5  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> ReferenceType
           decaf::lang::Pointer< T, REFCOUNTER >::operator*( ) const  [inline]
```

```
6.611.4.6  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> ReferenceType
           decaf::lang::Pointer< T, REFCOUNTER >::operator*( )  [inline]
```

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

### Returns

reference to the contained pointer.

### Exceptions

<code>NullPointerException</code>	if the contained value is Null
-----------------------------------	--------------------------------

```
6.611.4.7  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> PointerType
           decaf::lang::Pointer< T, REFCOUNTER >::operator->( )  [inline]
```

Indirection Operator, returns a pointer to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

### Returns

reference to the contained pointer.

### Exceptions

<code>NullPointerException</code>	if the contained value is Null
-----------------------------------	--------------------------------

```
6.611.4.8  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> PointerType
           decaf::lang::Pointer< T, REFCOUNTER >::operator->( ) const  [inline]
```

```
6.611.4.9  template<typename T, typename REFCOUNTER =
           decaf::util::concurrent::atomic::AtomicRefCount> Pointer&
           decaf::lang::Pointer< T, REFCOUNTER >::operator=( const Pointer< T,
           REFCOUNTER > & right ) throw ()  [inline]
```

Assigns the value of `right` to this **Pointer** (p. 2896) and increments the reference Count.

### Parameters

<i>right</i>	- <b>Pointer</b> (p. 2896) on the right hand side of an operator= call to this.
--------------	---

## 6.611 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference 2911

6.611.4.10 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 ,  
typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator=  
( const Pointer< T1, R1 > & right ) throw () [inline]`

6.611.4.11 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 ,  
typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator== (   
const Pointer< T1, R1 > & right ) const [inline]`

6.611.4.12 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> T* decaf::lang::Pointer< T,  
REFCOUNTER >::release ( ) [inline]`

Releases the **Pointer** (p. 2896) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 2896) is held by more than one object or this method is called from more than one thread.

### Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

### Returns

The pointer instance that was held by this **Pointer** (p. 2896) object, the pointer is no longer owned by this **Pointer** (p. 2896) and won't be freed when this **Pointer** (p. 2896) goes out of scope.

Referenced by `decaf::lang::Pointer< TransactionId >::Pointer()`.

6.611.4.13 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> void decaf::lang::Pointer<  
T, REFCOUNTER >::reset ( T * value ) [inline]`

Resets the **Pointer** (p. 2896) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2896) to a NULL pointer.

### Parameters

<i>value</i>	- The new value to contain.
--------------	-----------------------------

6.611.4.14 `template<typename T, typename REFCOUNTER =  
decaf::util::concurrent::atomic::AtomicRefCount> template<typename T1 >  
Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER  
>::staticCast ( ) const [inline]`

```
6.611.4.15 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> void decaf::lang::Pointer<
    T, REFCOUNTER >::swap ( Pointer< T, REFCOUNTER > & value ) throw ()
    [inline]
```

**Exception** (p. 1794) Safe Swap Function.

#### Parameters

<i>value</i>	- the value to swap with this.
--------------	--------------------------------

Referenced by decaf::lang::Pointer< TransactionId >::operator=(), and decaf::lang::Pointer< TransactionId >::swap().

### 6.611.5 Friends And Related Function Documentation

```
6.611.5.1 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> bool operator!= ( const
    Pointer< T, REFCOUNTER > & left, const T * right ) [friend]

6.611.5.2 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> bool operator!= ( const T * left,
    const Pointer< T, REFCOUNTER > & right ) [friend]

6.611.5.3 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> bool operator== ( const
    Pointer< T, REFCOUNTER > & left, const T * right ) [friend]

6.611.5.4 template<typename T, typename REFCOUNTER =
    decaf::util::concurrent::atomic::AtomicRefCount> bool operator== ( const T * left,
    const Pointer< T, REFCOUNTER > & right ) [friend]
```

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

### 6.612 decaf::lang::PointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2896) instance.

```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for decaf::lang::PointerComparator< T, R >:

## Public Member Functions

- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

### 6.612.1 Detailed Description

```
template<typename T, typename R = decaf::util::concurrent::atomic::AtomicRefCounter>class decaf::lang::PointerComparator<
T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2896) instance.

This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 2896) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

### 6.612.2 Member Function Documentation

```
6.612.2.1 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual int
decaf::lang::PointerComparator< T, R >::compare ( const Pointer< T, R > &
left, const Pointer< T, R > & right ) const [inline, virtual]
```

```
6.612.2.2 template<typename T , typename R =
decaf::util::concurrent::atomic::AtomicRefCounter> virtual bool
decaf::lang::PointerComparator< T, R >::operator() ( const Pointer< T, R >
& left, const Pointer< T, R > & right ) const [inline, virtual]
```

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

## 6.613 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

```
#include <src/main/activemq/cmsutil/PooledSession.h>
```

Inheritance diagram for activemq::cmsutil::PooledSession:

## Public Member Functions

- **PooledSession** (**SessionPool** \*pool, **cms::Session** \*session)
- virtual **~PooledSession** ()  
*Does nothing.*
- virtual **cms::Session** \* **getSession** ()  
*Returns a non-constant reference to the internal session object.*
- virtual const **cms::Session** \* **getSession** () const  
*Returns a constant reference to the internal session object.*
- virtual void **close** () throw ( cms::CMSEException )  
*Returns this session back to the pool, but does not close or destroy the internal session object.*
- virtual void **commit** () throw ( cms::CMSEException )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** () throw ( cms::CMSEException )  
*Rolls back all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** () throw ( cms::CMSEException )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual **cms::MessageConsumer** \* **createConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )  
*Creates a MessageConsumer for the specified destination, using a message selector.*
- virtual **cms::MessageConsumer** \* **createDurableConsumer** (const **cms::Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false) throw ( cms::CMSEException )  
*Creates a durable subscriber to the specified topic, using a Message selector.*
- virtual **cms::MessageConsumer** \* **createCachedConsumer** (const **cms::Destination** \*destination, const std::string &selector, bool noLocal) throw ( cms::CMSEException )  
*First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.*
- virtual **cms::MessageProducer** \* **createProducer** (const **cms::Destination** \*destination) throw ( cms::CMSEException )  
*Creates a MessageProducer to send messages to the specified destination.*



- virtual **cms::MessageProducer \* createCachedProducer** (const **cms::Destination** \*destination) throw ( cms::CMSExcption )  
*First checks the internal producer cache and creates one if none exist for the given destination.*
- virtual **cms::QueueBrowser \* createBrowser** (const **cms::Queue** \*queue) throw ( cms::CMSExcption )  
*Creates a new QueueBrowser to peek at Messages on the given Queue.*
- virtual **cms::QueueBrowser \* createBrowser** (const **cms::Queue** \*queue, const std::string &selector) throw ( cms::CMSExcption )  
*Creates a new QueueBrowser to peek at Messages on the given Queue.*
- virtual **cms::Queue \* createQueue** (const std::string &queueName) throw ( cms::CMSExcption )  
*Creates a queue identity given a Queue name.*
- virtual **cms::Topic \* createTopic** (const std::string &topicName) throw ( cms::CMSExcption )  
*Creates a topic identity given a Queue name.*
- virtual **cms::TemporaryQueue \* createTemporaryQueue** () throw ( cms::CMSExcption )  
*Creates a TemporaryQueue object.*
- virtual **cms::TemporaryTopic \* createTemporaryTopic** () throw ( cms::CMSExcption )  
*Creates a TemporaryTopic object.*
- virtual **cms::Message \* createMessage** () throw ( cms::CMSExcption )  
*Creates a new Message.*
- virtual **cms::BytesMessage \* createBytesMessage** () throw ( cms::CMSExcption )  
*Creates a BytesMessage.*
- virtual **cms::BytesMessage \* createBytesMessage** (const unsigned char \*bytes, int bytesSize) throw ( cms::CMSExcption )  
*Creates a BytesMessage and sets the payload to the passed value.*
- virtual **cms::StreamMessage \* createStreamMessage** () throw ( cms::CMSExcption )  
*Creates a new StreamMessage.*
- virtual **cms::TextMessage \* createTextMessage** () throw ( cms::CMSExcption )  
*Creates a new TextMessage.*
- virtual **cms::TextMessage \* createTextMessage** (const std::string &text) throw ( cms::CMSExcption )  
*Creates a new TextMessage and set the text to the value given.*
- virtual **cms::MapMessage \* createMapMessage** () throw ( cms::CMSExcption )  
*Creates a new MapMessage.*
- virtual **cms::Session::AcknowledgeMode getAcknowledgeMode** () const throw ( cms::CMSExcption )  
*Returns the acknowledgment mode of the session.*

- virtual bool **isTransacted** () const throw ( cms::CMSEException )  
*Gets if the Sessions is a Transacted Session.*
- virtual void **unsubscribe** (const std::string &name) throw ( cms::CMSEException )  
*Unsubscribes a durable subscription that has been created by a client.*

### Protected Member Functions

- **PooledSession** (const **PooledSession** &)
- **PooledSession & operator=** (const **PooledSession** &)

### 6.613.1 Detailed Description

A pooled session object that wraps around a delegate session.

Calls to close this session only result in giving the session back to the pool.

### 6.613.2 Constructor & Destructor Documentation

6.613.2.1 **activemq::cmsutil::PooledSession::PooledSession** ( const **PooledSession** & )  
[inline, protected]

6.613.2.2 **activemq::cmsutil::PooledSession::PooledSession** ( **SessionPool** \* *pool*,  
**cms::Session** \* *session* )

6.613.2.3 **virtual activemq::cmsutil::PooledSession::~~PooledSession** ( ) [virtual]

Does nothing.

### 6.613.3 Member Function Documentation

6.613.3.1 **virtual void activemq::cmsutil::PooledSession::close** ( ) throw ( **cms::CMSEException** ) [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object.

Implements **cms::Session** (p. 3309).

6.613.3.2 **virtual void activemq::cmsutil::PooledSession::commit** ( ) throw ( **cms::CMSEException** ) [inline, virtual]

Commits all messages done in this transaction and releases any locks currently held.

### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements **cms::Session** (p. 3309).

References cms::Session::commit().

**6.613.3.3** virtual **cms::QueueBrowser\*** activemq::cmsutil::PooledSession::createBrowser ( const cms::Queue \* *queue*, const std::string & *selector* ) throw ( cms::CMSEException ) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

#### Parameters

<i>queue</i>	the Queue to browse
<i>selector</i>	the Message selector to filter which messages are browsed.

#### Returns

New QueueBrowser that is owned by the caller.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3310).

**6.613.3.4** virtual **cms::QueueBrowser\*** activemq::cmsutil::PooledSession::createBrowser ( const cms::Queue \* *queue* ) throw ( cms::CMSEException ) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

#### Parameters

<i>queue</i>	the Queue to browse
--------------	---------------------

#### Returns

New QueueBrowser that is owned by the caller.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if the destination given is invalid.

Implements **cms::Session** (p. 3309).

```
6.613.3.5 virtual cms::BytesMessage* ac-
          tivemq::cmsutil::PooledSession::createBytesMessage ( ) throw
          ( cms::CMSEException) [inline, virtual]
```

Creates a BytesMessage.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3310).

```
6.613.3.6 virtual cms::BytesMessage* ac-
          tivemq::cmsutil::PooledSession::createBytesMessage ( const
          unsigned char * bytes, int bytesSize ) throw ( cms::CMSEException)
          [inline, virtual]
```

Creates a BytesMessage and sets the payload to the passed value.

#### Parameters

<i>bytes</i>	an array of bytes to set in the message
<i>bytesSize</i>	the size of the bytes array, or number of bytes to use

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3311).

```
6.613.3.7 virtual cms::MessageConsumer* ac-
          tivemq::cmsutil::PooledSession::createCachedConsumer ( const
          cms::Destination * destination, const std::string & selector, bool noLocal ) throw
          ( cms::CMSEException) [virtual]
```

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.

If created, the consumer is added to the pool's lifecycle manager.

#### Parameters

<i>destination</i>	the destination to receive on
<i>selector</i>	the selector to use
<i>noLocal</i>	whether or not to receive messages from the same connection

**Returns**

the consumer resource

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	if something goes wrong.
--	--------------------------

```
6.613.3.8 virtual cms::MessageProducer* ac-
tivemq::cmsutil::PooledSession::createCachedProducer ( const
cms::Destination * destination ) throw ( cms::CMSException )
[virtual]
```

First checks the internal producer cache and creates one if none exist for the given destination.

If created, the producer is added to the pool's lifecycle manager.

**Parameters**

<i>destination</i>	the destination to send on
--------------------	----------------------------

**Returns**

the producer resource

**Exceptions**

<b><i>cms::CMSException</i></b> (p. 1130)	if something goes wrong.
--	--------------------------

```
6.613.3.9 virtual cms::MessageConsumer* ac-
tivemq::cmsutil::PooledSession::createConsumer ( const
cms::Destination * destination ) throw ( cms::CMSException ) [inline,
virtual]
```

Creates a MessageConsumer for the specified destination.

**Parameters**

<i>destination</i>	the Destination that this consumer receiving messages for.
--------------------	--

**Returns**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 3311).

6.613.3.10 **virtual cms::MessageConsumer\* activemq::cmsutil::PooledSession::createConsumer ( const cms::Destination \* *destination*, const std::string & *selector* ) throw ( cms::CMSEException )** *[inline, virtual]*

Creates a MessageConsumer for the specified destination, using a message selector.

**Parameters**

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use

**Returns**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.
<i>InvalidSelectorException</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 3311).

6.613.3.11 **virtual cms::MessageConsumer\* activemq::cmsutil::PooledSession::createConsumer ( const cms::Destination \* *destination*, const std::string & *selector*, bool *noLocal* ) throw ( cms::CMSEException )** *[inline, virtual]*

Creates a MessageConsumer for the specified destination, using a message selector.

**Parameters**

<i>destination</i>	the Destination that this consumer receiving messages for.
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns**

pointer to a new MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions**

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.
<i>InvalidSelectorException</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 3312).

```
6.613.3.12 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createDurableConsumer (
    const cms::Topic * destination, const std::string & name, const std::string &
    selector, bool noLocal = false ) throw ( cms::CMSException ) [inline,
    virtual]
```

Creates a durable subscriber to the specified topic, using a Message selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

**Parameters**

<i>destination</i>	the topic to subscribe to
<i>name</i>	The name used to identify the subscription
<i>selector</i>	the Message Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns**

pointer to a new durable MessageConsumer that is owned by the caller ( caller deletes )

**Exceptions**

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.
<i>InvalidSelectorException</i>	- if the message selector is invalid.

Implements **cms::Session** (p. 3313).

6.613.3.13 `virtual cms::MapMessage* activemq::cmsutil::PooledSession::createMapMessage ( ) throw ( cms::CMSException ) [inline, virtual]`

Creates a new MapMessage.

#### Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::Session** (p. 3314).

6.613.3.14 `virtual cms::Message* activemq::cmsutil::PooledSession::createMessage ( ) throw ( cms::CMSException ) [inline, virtual]`

Creates a new Message.

#### Exceptions

<i>CMSException</i>	- If an internal error occurs.
---------------------	--------------------------------

Implements **cms::Session** (p. 3314).

6.613.3.15 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer ( const cms::Destination * destination ) throw ( cms::CMSException ) [inline, virtual]`

Creates a MessageProducer to send messages to the specified destination.

#### Parameters

<i>destination</i>	the Destination to send on
--------------------	----------------------------

#### Returns

New MessageProducer that is owned by the caller.

#### Exceptions

<i>CMSException</i>	- If an internal error occurs.
<i>InvalidDestinationException</i>	- if an invalid destination is specified.

Implements **cms::Session** (p. 3314).



```
6.613.3.16 virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue ( const
std::string & queueName ) throw ( cms::CMSEException ) [inline,
virtual]
```

Creates a queue identity given a Queue name.

#### Parameters

<i>queueName</i>	the name of the new Queue
------------------	---------------------------

#### Returns

new Queue pointer that is owned by the caller.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3315).

```
6.613.3.17 virtual cms::StreamMessage* ac-
tivemq::cmsutil::PooledSession::createStreamMessage ( )
throw ( cms::CMSEException ) [inline, virtual]
```

Creates a new StreamMessage.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3315).

```
6.613.3.18 virtual cms::TemporaryQueue* ac-
tivemq::cmsutil::PooledSession::createTemporaryQueue ( )
throw ( cms::CMSEException ) [inline, virtual]
```

Creates a TemporaryQueue object.

#### Returns

new TemporaryQueue pointer that is owned by the caller.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3315).

```
6.613.3.19 virtual cms::TemporaryTopic* ac-
          tivemq::cmsutil::PooledSession::createTemporaryTopic ( )
          throw ( cms::CMSEException ) [inline, virtual]
```

Creates a TemporaryTopic object.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3316).

```
6.613.3.20 virtual cms::TextMessage* ac-
          tivemq::cmsutil::PooledSession::createTextMessage ( const
          std::string & text ) throw ( cms::CMSEException ) [inline, virtual]
```

Creates a new TextMessage and set the text to the value given.

#### Parameters

<i>text</i>	the initial text for the message
-------------	----------------------------------

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3316).

```
6.613.3.21 virtual cms::TextMessage* ac-
          tivemq::cmsutil::PooledSession::createTextMessage ( ) throw
          ( cms::CMSEException ) [inline, virtual]
```

Creates a new TextMessage.

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3316).

```
6.613.3.22 virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic ( const
          std::string & topicName ) throw ( cms::CMSEException ) [inline,
          virtual]
```

Creates a topic identity given a Queue name.

#### Parameters

<i>topicName</i>	the name of the new Topic
------------------	---------------------------

**Returns**

new Topic pointer that is owned by the caller.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3317).

**6.613.3.23** `virtual cms::Session::AcknowledgeMode  
activemq::cmsutil::PooledSession::getAcknowledgeMode ( ) const throw (  
cms::CMSEException) [inline, virtual]`

Returns the acknowledgment mode of the session.

**Returns**

the Sessions Acknowledge Mode

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3317).

**6.613.3.24** `virtual cms::Session* activemq::cmsutil::PooledSession::getSession ( )  
[inline, virtual]`

Returns a non-constant reference to the internal session object.

**Returns**

the session object.

**6.613.3.25** `virtual const cms::Session* activemq::cmsutil::PooledSession::getSession ( )  
const [inline, virtual]`

Returns a constant reference to the internal session object.

**Returns**

the session object.

**6.613.3.26** `virtual bool activemq::cmsutil::PooledSession::isTransacted ( ) const throw (  
cms::CMSEException) [inline, virtual]`

Gets if the Sessions is a Transacted Session.

**Returns**

transacted true - false.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3317).

6.613.3.27 **PooledSession& activemq::cmsutil::PooledSession::operator= ( const PooledSession & )** [*inline, protected*]

6.613.3.28 **virtual void activemq::cmsutil::PooledSession::recover ( ) throw ( cms::CMSEException )** [*inline, virtual*]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

**Exceptions**

<i>CMSEException</i>	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<i>IllegalStateException</i>	- if the method is called by a transacted session.

Implements **cms::Session** (p. 3318).

6.613.3.29 **virtual void activemq::cmsutil::PooledSession::rollback ( ) throw ( cms::CMSEException )** [*inline, virtual*]

Rolls back all messages done in this transaction and releases any locks currently held.

**Exceptions**

<i>CMSEException</i>	- If an internal error occurs.
<i>IllegalStateException</i>	- if the method is not called by a transacted session.

Implements **cms::Session** (p. 3318).

6.613.3.30 virtual void activemq::cmsutil::PooledSession::unsubscribe ( const std::string & *name* ) throw ( cms::CMSEException ) [inline, virtual]

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

#### Parameters

<i>name</i>	The name used to identify this subscription
-------------	---

#### Exceptions

<i>CMSEException</i>	- If an internal error occurs.
----------------------	--------------------------------

Implements **cms::Session** (p. 3319).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

## 6.614 decaf::util::concurrent::PooledThread Class Reference

```
#include <src/main/decaf/util/concurrent/PooledThread.h>
```

Inheritance diagram for decaf::util::concurrent::PooledThread:

#### Public Member Functions

- **PooledThread** (**ThreadPool** \*pool)  
*Constructor.*
- virtual ~**PooledThread** ()
- virtual void **run** ()  
*Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3718) and then grabs it and calls its run method.*
- virtual void **stop** () throw ( lang::Exception )  
*Stops the Thread, thread will complete its task if currently running one, and then die.*
- virtual bool **isBusy** ()  
*Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.*

- virtual void **setPooledThreadListener** (**PooledThreadListener** \*listener)  
*Adds a listener to this **PooledThread** (p. 2918) to be notified when this thread starts and completes a task.*
- virtual **PooledThreadListener** \* **getPooledThreadListener** ()  
*Removes a listener for this **PooledThread** (p. 2918) to be notified when this thread starts and completes a task.*

### 6.614.1 Constructor & Destructor Documentation

#### 6.614.1.1 `decaf::util::concurrent::PooledThread::PooledThread ( ThreadPool * pool )`

Constructor.

##### Parameters

<code>pool</code>	the parant <b>ThreadPool</b> (p. 3718) object
-------------------	---

#### 6.614.1.2 `virtual decaf::util::concurrent::PooledThread::~~PooledThread ( ) [virtual]`

### 6.614.2 Member Function Documentation

#### 6.614.2.1 `virtual PooledThreadListener* decaf::util::concurrent::PooledThread::getPooledThreadListener ( ) [inline, virtual]`

Removes a listener for this **PooledThread** (p. 2918) to be notified when this thread starts and completes a task.

##### Returns

a pointer to this thread's listener or NULL

#### 6.614.2.2 `virtual bool decaf::util::concurrent::PooledThread::isBusy ( ) [inline, virtual]`

Checks to see if the thread is busy, if busy it means that this thread has taken a task from the **ThreadPool**'s queue and is processing it.

##### Returns

true if the Thread is busy

#### 6.614.2.3 `virtual void decaf::util::concurrent::PooledThread::run ( ) [virtual]`

Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 3718) and then grabs it and calls its run method.

Reimplemented from **decaf::lang::Thread** (p. 3713).

6.614.2.4 `virtual void decaf::util::concurrent::PooledThread::setPooledThreadListener ( PooledThreadListener * listener ) [inline, virtual]`

Adds a listener to this **PooledThread** (p. 2918) to be notified when this thread starts and completes a task.

#### Parameters

<i>listener</i>	the listener to send notifications to.
-----------------	--

6.614.2.5 `virtual void decaf::util::concurrent::PooledThread::stop ( ) throw ( lang::Exception ) [virtual]`

Stops the Thread, thread will complete its task if currently running one, and then die.

Does not block.

#### Exceptions

<i>Exception</i>	
------------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThread.h**

## 6.615 decaf::util::concurrent::PooledThreadListener Class Reference

Abstract Listener Interface for users of **ThreadPool** (p. 3718).

```
#include <src/main/decaf/util/concurrent/PooledThreadListener.h>
```

Inheritance diagram for decaf::util::concurrent::PooledThreadListener:

#### Public Member Functions

- virtual `~PooledThreadListener ()`
- virtual void `onTaskStarted (PooledThread *thread)=0`  
*Called by a pooled thread when it is about to begin executing a new task.*
- virtual void `onTaskCompleted (PooledThread *thread)=0`  
*Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.*
- virtual void `onTaskException (PooledThread *thread, lang::Exception &ex)=0`

*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2918) is now no longer running.*

### 6.615.1 Detailed Description

Abstract Listener Interface for users of **ThreadPool** (p. 3718).

The implementor of this class receives events related to the execution and termination of threads running in the **ThreadPool** (p. 3718).

#### Since

1.0

### 6.615.2 Constructor & Destructor Documentation

6.615.2.1 `virtual decaf::util::concurrent::PooledThreadListener::~~PooledThreadListener ( )`  
[inline, virtual]

### 6.615.3 Member Function Documentation

6.615.3.1 `virtual void decaf::util::concurrent::PooledThreadListener::onTaskCompleted ( PooledThread * thread )` [pure virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

#### Parameters

<i>thread</i>	- Pointer the the Pooled Thread that is making this call.
---------------	---

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3722).

6.615.3.2 `virtual void decaf::util::concurrent::PooledThreadListener::onTaskException ( PooledThread * thread, lang::Exception & ex )` [pure virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2918) is now no longer running.

#### Parameters

<i>thread</i>	- Pointer to the Pooled Thread that is making this call
<i>ex</i>	- The Exception that occurred.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3722).



6.615.3.3 virtual void decaf::util::concurrent::PooledThreadListener::onTaskStarted ( PooledThread \* thread ) [pure virtual]

Called by a pooled thread when it is about to begin executing a new task.

#### Parameters

<i>thread</i>	- Pointer to the Pooled Thread that is making this call
---------------	---

Implemented in **decaf::util::concurrent::ThreadPool** (p. 3722).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThreadListener.h**

## 6.616 decaf::net::PortUnreachableException Class Reference

```
#include <src/main/decaf/net/PortUnreachableException.h>
```

Inheritance diagram for decaf::net::PortUnreachableException:

### Public Member Functions

- **PortUnreachableException** () throw ()  
*Default Constructor.*
- **PortUnreachableException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()  
*Copy Constructor.*
- **PortUnreachableException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **PortUnreachableException** (const std::exception \*cause) throw ()  
*Constructor.*
- **PortUnreachableException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **PortUnreachableException** \* **clone** () const  
*Clones this exception.*
- virtual ~**PortUnreachableException** () throw ()

### 6.616.1 Constructor & Destructor Documentation

6.616.1.1 `decaf::net::PortUnreachableException::PortUnreachableException ( ) throw ()`  
`[inline]`

Default Constructor.

6.616.1.2 `decaf::net::PortUnreachableException::PortUnreachableException ( const Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other Exception.

#### Parameters

<code>ex</code>	An exception that should become this type of Exception
-----------------	--

6.616.1.3 `decaf::net::PortUnreachableException::PortUnreachableException ( const PortUnreachableException & ex ) throw ()` `[inline]`

Copy Constructor.

#### Parameters

<code>ex</code>	An exception that should become this type of Exception
-----------------	--

6.616.1.4 `decaf::net::PortUnreachableException::PortUnreachableException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<code>file</code>	The file name where exception occurs
<code>lineNumber</code>	The line number where the exception occurred.
<code>cause</code>	The exception that was the cause for this one to be thrown.
<code>msg</code>	The message to report
<code>...</code>	list of primitives that are formatted into the message

6.616.1.5 `decaf::net::PortUnreachableException::PortUnreachableException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.616.1.6 `decaf::net::PortUnreachableException::PortUnreachableException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.616.1.7 `virtual decaf::net::PortUnreachableException::~~PortUnreachableException ( ) throw () [inline, virtual]`

**6.616.2 Member Function Documentation**

6.616.2.1 `virtual PortUnreachableException* decaf::net::PortUnreachableException::clone ( )const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 3467).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/PortUnreachableException.h`

**6.617 activemq::core::PrefetchPolicy Class Reference**

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

```
#include <src/main/activemq/core/PrefetchPolicy.h>
```

Inheritance diagram for `activemq::core::PrefetchPolicy`:

## Public Member Functions

- virtual `~PrefetchPolicy ()`
- virtual void **setDurableTopicPrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Durable Topic.*
- virtual int **getDurableTopicPrefetch** () const =0  
*Gets the amount of messages to prefetch for a Durable Topic.*
- virtual void **setQueuePrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Queue.*
- virtual int **getQueuePrefetch** () const =0  
*Gets the amount of messages to prefetch for a Queue.*
- virtual void **setQueueBrowserPrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Queue Browser.*
- virtual int **getQueueBrowserPrefetch** () const =0  
*Gets the amount of messages to prefetch for a Queue Browser.*
- virtual void **setTopicPrefetch** (int value)=0  
*Sets the amount of prefetched messages for a Topic.*
- virtual int **getTopicPrefetch** () const =0  
*Gets the amount of messages to prefetch for a Topic.*
- virtual int **getMaxPrefetchLimit** (int value) const =0  
*Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.*
- virtual **PrefetchPolicy \* clone** () const =0  
*Clone the Policy and return a new pointer to that clone.*
- virtual void **configure** (const **decaf::util::Properties** &properties)  
*Checks the supplied properties object for properties matching the configurable settings of this class.*

## Protected Member Functions

- **PrefetchPolicy** ()

### 6.617.1 Detailed Description

Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.

Since

3.2.0

## 6.617.2 Constructor & Destructor Documentation

6.617.2.1 `activemq::core::PrefetchPolicy::PrefetchPolicy ( )` `[protected]`

6.617.2.2 `virtual activemq::core::PrefetchPolicy::~~PrefetchPolicy ( )` `[virtual]`

## 6.617.3 Member Function Documentation

6.617.3.1 `virtual PrefetchPolicy* activemq::core::PrefetchPolicy::clone ( ) const` `[pure virtual]`

Clone the Policy and return a new pointer to that clone.

### Returns

pointer to a new **PrefetchPolicy** (p. 2924) instance that is a clone of this one.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1641).

6.617.3.2 `virtual void activemq::core::PrefetchPolicy::configure ( const decaf::util::Properties & properties )` `[virtual]`

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.PrefetchPolicy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.PrefetchPolicy.topicPrefetch` will be used to set the value of the topic prefetch limit.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

### Parameters

<i>properties</i>	The Properties object used to configure this object.
-------------------	--

### Exceptions

<i>NumberFormatException</i>	if a property that is numeric cannot be converted
<i>IllegalArgumentException</i>	if a property can't be converted to the correct type.

6.617.3.3 `virtual int activemq::core::PrefetchPolicy::getDurableTopicPrefetch ( ) const` `[pure virtual]`

Gets the amount of messages to prefetch for a Durable Topic.

**Returns**

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1641).

```
6.617.3.4  virtual int activemq::core::PrefetchPolicy::getMaxPrefetchLimit ( int value ) const  
          [pure virtual]
```

Given a requested value for a new prefetch limit, compare it against some max prefetch value and return either the requested value or the maximum allowable value for prefetch.

**Returns**

the allowable value for a prefetch limit, either requested or the max.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1642).

```
6.617.3.5  virtual int activemq::core::PrefetchPolicy::getQueueBrowserPrefetch ( ) const  
          [pure virtual]
```

Gets the amount of messages to prefetch for a Queue Browser.

**Returns**

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1642).

```
6.617.3.6  virtual int activemq::core::PrefetchPolicy::getQueuePrefetch ( ) const  [pure  
          virtual]
```

Gets the amount of messages to prefetch for a Queue.

**Returns**

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1642).

```
6.617.3.7  virtual int activemq::core::PrefetchPolicy::getTopicPrefetch ( ) const  [pure  
          virtual]
```

Gets the amount of messages to prefetch for a Topic.

**Returns**

value of the number of messages to prefetch.

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1642).

**6.617.3.8** `virtual void activemq::core::PrefetchPolicy::setDurableTopicPrefetch ( int value )`  
[pure virtual]

Sets the amount of prefetched messages for a Durable Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1643).

**6.617.3.9** `virtual void activemq::core::PrefetchPolicy::setQueueBrowserPrefetch ( int value )`  
[pure virtual]

Sets the amount of prefetched messages for a Queue Browser.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1643).

**6.617.3.10** `virtual void activemq::core::PrefetchPolicy::setQueuePrefetch ( int value )` [pure virtual]

Sets the amount of prefetched messages for a Queue.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1643).

**6.617.3.11** `virtual void activemq::core::PrefetchPolicy::setTopicPrefetch ( int value )` [pure virtual]

Sets the amount of prefetched messages for a Topic.

#### Parameters

<i>value</i>	The number of messages to prefetch.
--------------	-------------------------------------

Implemented in **activemq::core::policies::DefaultPrefetchPolicy** (p. 1643).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**PrefetchPolicy.h**

## 6.618 activemq::util::PrimitiveList Class Reference

List of primitives.

```
#include <src/main/activemq/util/PrimitiveList.h>
```

Inheritance diagram for activemq::util::PrimitiveList:

### Public Member Functions

- **PrimitiveList** ()  
*Default Constructor, creates an Empty list.*
- virtual **~PrimitiveList** ()
- **PrimitiveList** (const **decaf::util::List**< **PrimitiveValueNode** > &src)  
*Copy Constructor.*
- **PrimitiveList** (const **PrimitiveList** &src)  
*Copy Constructor.*
- std::string **toString** () const  
*Converts the contents into a formatted string that can be output in a Log File or other debugging tool.*
- virtual bool **getBool** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Boolean value at the specified index.*
- virtual void **setBool** (std::size\_t index, bool value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual unsigned char **getByte** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Byte value at the specified index.*
- virtual void **setByte** (std::size\_t index, unsigned char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual char **getChar** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Character value at the specified index.*
- virtual void **setChar** (std::size\_t index, char value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*



- virtual short **getShort** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Short value at the specified index.*
- virtual void **setShort** (std::size\_t index, short value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual int **getInt** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Integer value at the specified index.*
- virtual void **setInt** (std::size\_t index, int value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual long long **getLong** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Long value at the specified index.*
- virtual void **setLong** (std::size\_t index, long long value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual float **getFloat** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Float value at the specified index.*
- virtual void **setFloat** (std::size\_t index, float value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual double **getDouble** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Double value at the specified index.*
- virtual void **setDouble** (std::size\_t index, double value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*
- virtual std::string **getString** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the String value at the specified index.*
- virtual void **setString** (std::size\_t index, const std::string &value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

- virtual `std::vector< unsigned char > getByteArray (std::size_t index) const` throw ( `decaf::lang::exceptions::IndexOutOfBoundsException`, `decaf::lang::exceptions::UnsupportedOperationException` )

*Gets the Byte Array value at the specified index.*

- virtual void `setByteArray (std::size_t index, const std::vector< unsigned char > &value) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`

*Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.*

### 6.618.1 Detailed Description

List of primitives.

### 6.618.2 Constructor & Destructor Documentation

#### 6.618.2.1 `activemq::util::PrimitiveList::PrimitiveList ( )`

Default Constructor, creates an Empty list.

#### 6.618.2.2 `virtual activemq::util::PrimitiveList::~~PrimitiveList ( )` [virtual]

#### 6.618.2.3 `activemq::util::PrimitiveList::PrimitiveList ( const decaf::util::List< PrimitiveValueNode > & src )`

Copy Constructor.

##### Parameters

<code>src</code>	- the Decaf List of PrimitiveNodeValues to copy
------------------	---

#### 6.618.2.4 `activemq::util::PrimitiveList::PrimitiveList ( const PrimitiveList & src )`

Copy Constructor.

##### Parameters

<code>src</code>	- the <b>PrimitiveList</b> (p. 2929) to copy
------------------	--

### 6.618.3 Member Function Documentation

```
6.618.3.1 virtual bool activemq::util::PrimitiveList::getBool ( std::size_t index ) const
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
        decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Gets the Boolean value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

```
6.618.3.2 virtual unsigned char activemq::util::PrimitiveList::getBytes ( std::size_t index )
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
              decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Gets the Byte value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.618.3.3 `virtual std::vector<unsigned char> activemq::util::PrimitiveList::getByteArray  
( std::size_t index ) const throw ( de-  
caf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Byte Array value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.618.3.4 `virtual char activemq::util::PrimitiveList::getChar ( std::size_t index ) const  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Character value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

```
6.618.3.5 virtual double activemq::util::PrimitiveList::getDouble ( std::size_t index )  
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]
```

Gets the Double value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

```
6.618.3.6 virtual float activemq::util::PrimitiveList::getFloat ( std::size_t index ) const  
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]
```

Gets the Float value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

```
6.618.3.7 virtual int activemq::util::PrimitiveList::getInt ( std::size_t index ) const
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
        decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Gets the Integer value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

```
6.618.3.8 virtual long long activemq::util::PrimitiveList::getLong ( std::size_t index )
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
              decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Gets the Long value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

```
6.618.3.9 virtual short activemq::util::PrimitiveList::getShort ( std::size_t index ) const
throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Gets the Short value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

```
6.618.3.10 virtual std::string activemq::util::PrimitiveList::getString ( std::size_t index )
const throw ( decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Gets the String value at the specified index.

#### Parameters

<i>index</i>	- index to get value from
--------------	---------------------------

#### Returns

value contained at the given index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is > <b>size()</b> (p. 3542)
<i>UnsupportedOperationException</i>	if the type at index is not of the type that this method is to return or can convert to.

6.618.3.11 `virtual void activemq::util::PrimitiveList::setBool ( std::size_t index, bool value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

6.618.3.12 `virtual void activemq::util::PrimitiveList::setByte ( std::size_t index, unsigned char value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )`  
`[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

6.618.3.13 `virtual void activemq::util::PrimitiveList::setByteArray ( std::size_t index, const std::vector< unsigned char > & value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )` `[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index



### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

**6.618.3.14** virtual void activemq::util::PrimitiveList::setChar ( std::size\_t *index*, char *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

**6.618.3.15** virtual void activemq::util::PrimitiveList::setDouble ( std::size\_t *index*, double *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

**6.618.3.16** virtual void activemq::util::PrimitiveList::setFloat ( std::size\_t *index*, float *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the

index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

6.618.3.17 `virtual void activemq::util::PrimitiveList::setInt ( std::size_t index, int value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

6.618.3.18 `virtual void activemq::util::PrimitiveList::setLong ( std::size_t index, long long value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

6.618.3.19 virtual void activemq::util::PrimitiveList::setShort ( std::size\_t *index*, short *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
[virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

6.618.3.20 virtual void activemq::util::PrimitiveList::setString ( std::size\_t *index*, const std::string & *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

#### Parameters

<i>index</i>	- location to set in the list
<i>value</i>	- the new value to assign to the element at index

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index > <b>size()</b> (p. 3542).
----------------------------------	-------------------------------------

6.618.3.21 std::string activemq::util::PrimitiveList::toString ( ) const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

#### Returns

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveList.h**

## 6.619 activemq::util::PrimitiveMap Class Reference

Map of named primitives.

```
#include <src/main/activemq/util/PrimitiveMap.h>
```

Inheritance diagram for activemq::util::PrimitiveMap:

### Public Member Functions

- **PrimitiveMap** ()

*Default Constructor, creates an empty map.*

- virtual **~PrimitiveMap** ()

- **PrimitiveMap** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &source)

*Copy Constructor.*

- **PrimitiveMap** (const **PrimitiveMap** &source)

*Copy Constructor.*

- std::string **toString** () const

*Converts the contents into a formatted string that can be output in a Log File or other debugging tool.*

- virtual bool **getBool** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setBool** (const std::string &key, bool value)

*Sets the value at key to the specified type.*

- virtual unsigned char **getByte** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setByte** (const std::string &key, unsigned char value)

*Sets the value at key to the specified type.*

- virtual char **getChar** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setChar** (const std::string &key, char value)

*Sets the value at key to the specified type.*

- virtual short **getShort** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException decaf::lang::exceptions::UnsupportedOperationException )

*Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*

- virtual void **setShort** (const std::string &key, short value)  
*Sets the value at key to the specified type.*
- virtual int **getInt** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setInt** (const std::string &key, int value)  
*Sets the value at key to the specified type.*
- virtual long long **getLong** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setLong** (const std::string &key, long long value)  
*Sets the value at key to the specified type.*
- virtual float **getFloat** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setFloat** (const std::string &key, float value)  
*Sets the value at key to the specified type.*
- virtual double **getDouble** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setDouble** (const std::string &key, double value)  
*Sets the value at key to the specified type.*
- virtual std::string **getString** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setString** (const std::string &key, const std::string &value)  
*Sets the value at key to the specified type.*
- virtual std::vector< unsigned char > **getByteArray** (const std::string &key) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
*Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.*
- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned char > &value)  
*Sets the value at key to the specified type.*

### 6.619.1 Detailed Description

Map of named primitives.

## 6.619.2 Constructor & Destructor Documentation

### 6.619.2.1 `activemq::util::PrimitiveMap::PrimitiveMap ( )`

Default Constructor, creates an empty map.

### 6.619.2.2 `virtual activemq::util::PrimitiveMap::~~PrimitiveMap ( ) [virtual]`

### 6.619.2.3 `activemq::util::PrimitiveMap::PrimitiveMap ( const decaf::util::Map< std::string, PrimitiveValueNode > & source )`

Copy Constructor.

#### Parameters

<i>source</i>	The Decaf Library Map instance whose elements will be copied into this Map.
---------------	---

### 6.619.2.4 `activemq::util::PrimitiveMap::PrimitiveMap ( const PrimitiveMap & source )`

Copy Constructor.

#### Parameters

<i>source</i>	The <b>PrimitiveMap</b> (p. 2941) whose elements will be copied into this Map.
---------------	--

## 6.619.3 Member Function Documentation

### 6.619.3.1 `virtual bool activemq::util::PrimitiveMap::getBool ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
-------------------------------	---------------------------

<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns
--------------------------------------	--

**6.619.3.2** virtual unsigned char activemq::util::PrimitiveMap::getByte ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

**6.619.3.3** virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getByteArray ( const std::string & key ) const throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
-------------------------------	---------------------------

<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns
--------------------------------------	--

6.619.3.4 `virtual char activemq::util::PrimitiveMap::getChar ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.619.3.5 `virtual double activemq::util::PrimitiveMap::getDouble ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns



```
6.619.3.6 virtual float activemq::util::PrimitiveMap::getFloat ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]
```

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

```
6.619.3.7 virtual int activemq::util::PrimitiveMap::getInt ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]
```

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.619.3.8 `virtual long long activemq::util::PrimitiveMap::getLong ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

6.619.3.9 `virtual short activemq::util::PrimitiveMap::getShort ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]`

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

```
6.619.3.10 virtual std::string activemq::util::PrimitiveMap::getString ( const std::string & key )  
const throw ( decaf::lang::exceptions::NoSuchElementException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[virtual]
```

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

#### Parameters

<i>key</i>	- the location to return the value from.
------------	--

#### Returns

the value at key in the type requested.

#### Exceptions

<i>NoSuchElementException</i>	if key is not in the map.
<i>UnsupportedOperationException</i>	if the value cannot be converted to the type this method returns

```
6.619.3.11 virtual void activemq::util::PrimitiveMap::setBool ( const std::string & key, bool  
value ) [virtual]
```

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

```
6.619.3.12 virtual void activemq::util::PrimitiveMap::setByte ( const std::string & key, unsigned  
char value ) [virtual]
```

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.13** `virtual void activemq::util::PrimitiveMap::setByteArray ( const std::string & key,  
const std::vector< unsigned char > & value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.14** `virtual void activemq::util::PrimitiveMap::setChar ( const std::string & key, char  
value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.15** `virtual void activemq::util::PrimitiveMap::setDouble ( const std::string & key, double  
value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

**6.619.3.16** `virtual void activemq::util::PrimitiveMap::setFloat ( const std::string & key, float  
value ) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.619.3.17 `virtual void activemq::util::PrimitiveMap::setInt ( const std::string & key, int value )`  
[virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.619.3.18 `virtual void activemq::util::PrimitiveMap::setLong ( const std::string & key, long long value )` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.619.3.19 `virtual void activemq::util::PrimitiveMap::setShort ( const std::string & key, short value )` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.619.3.20 `virtual void activemq::util::PrimitiveMap::setString ( const std::string & key, const std::string & value )` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

#### Parameters

<i>key</i>	- the map key to set or insert.
<i>value</i>	- the new value to set at the key location.

6.619.3.21 `std::string activemq::util::PrimitiveMap::toString ( ) const`

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

#### Returns

formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveMap.h`

## 6.620 `activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller` Class Reference

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

```
#include <src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h>
```

#### Public Member Functions

- `PrimitiveTypesMarshaller ( )`
- `virtual ~PrimitiveTypesMarshaller ( )`

#### Static Public Member Functions

- static void **marshal** (const `util::PrimitiveMap` \*map, `std::vector< unsigned char >` &buffer) throw ( `decaf::lang::Exception` )  
*Marshal a primitive map object to the given byte buffer.*
- static void **unmarshal** (`util::PrimitiveMap` \*map, const `std::vector< unsigned char >` &buffer) throw ( `decaf::lang::Exception` )  
*Unmarshal a PrimitiveMap from the provided Byte buffer.*
- static void **marshal** (const `util::PrimitiveList` \*list, `std::vector< unsigned char >` &buffer) throw ( `decaf::lang::Exception` )  
*Marshal a primitive list object to the given byte buffer.*
- static void **unmarshal** (`util::PrimitiveList` \*list, const `std::vector< unsigned char >` &buffer) throw ( `decaf::lang::Exception` )  
*Unmarshal a PrimitiveList from the provided byte buffer.*
- static void **marshalMap** (const `util::PrimitiveMap` \*map, `decaf::io::DataOutputStream` &dataOut) throw ( `decaf::lang::Exception` )  
*Marshal a primitive map object to the given DataOutputStream.*
- static `util::PrimitiveMap` \* **unmarshalMap** (`decaf::io::DataInputStream` &dataIn) throw ( `decaf::lang::Exception` )

*Unmarshal a PrimitiveMap from the provided DataInputStream.*

- static void **marshalList** (const **util::PrimitiveList** \*list, **decaf::io::DataOutputStream** &dataOut) throw ( **decaf::lang::Exception** )

*Marshal a PrimitiveList to the given DataOutputStream.*

- static **util::PrimitiveList** \* **unmarshalList** (**decaf::io::DataInputStream** &dataIn) throw ( **decaf::lang::Exception** )

*Unmarshal a PrimitiveList from the given DataInputStream.*

## Static Protected Member Functions

- static void **marshalPrimitiveMap** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::Map**< std::string, **util::PrimitiveValueNode** > &map) throw ( **decaf::io::IOException** )

*Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.*

- static void **marshalPrimitiveList** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::List**< **util::PrimitiveValueNode** > &list) throw ( **decaf::io::IOException** )

*Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.*

- static void **marshalPrimitive** (**decaf::io::DataOutputStream** &dataOut, const **util::PrimitiveValueNode** &value) throw ( **decaf::io::IOException** )

*Used to Marshal the Primitive types out on the Wire.*

- static void **unmarshalPrimitiveMap** (**decaf::io::DataInputStream** &dataIn, **util::PrimitiveMap** &map) throw ( **decaf::io::IOException** )

*Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.*

- static void **unmarshalPrimitiveList** (**decaf::io::DataInputStream** &dataIn, **decaf::util::StlList**< **util::PrimitiveValueNode** > &list) throw ( **decaf::io::IOException** )

*Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.*

- static **util::PrimitiveValueNode** **unmarshalPrimitive** (**decaf::io::DataInputStream** &dataIn) throw ( **decaf::io::IOException** )

*Unmarshals a Primitive Type from the stream, and returns it as a value Node.*

## 6.620.1 Detailed Description

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

## 6.620.2 Constructor & Destructor Documentation

- 6.620.2.1 **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller**  
( ) `[inline]`

6.620.2.2 `virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~~PrimitiveTypesMarshaller ( ) [inline, virtual]`

### 6.620.3 Member Function Documentation

6.620.3.1 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal ( const util::PrimitiveMap * map, std::vector< unsigned char > & buffer ) throw ( decaf::lang::Exception ) [static]`

Marshal a primitive map object to the given byte buffer.

#### Parameters

<i>map</i>	Map to Marshal.
<i>buffer</i>	The byte buffer to write the marshaled data to.

#### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

6.620.3.2 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal ( const util::PrimitiveList * list, std::vector< unsigned char > & buffer ) throw ( decaf::lang::Exception ) [static]`

Marshal a primitive list object to the given byte buffer.

#### Parameters

<i>map</i>	The PrimitiveList to Marshal.
<i>buffer</i>	The byte buffer to write the marshaled data to.

#### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

6.620.3.3 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalList ( const util::PrimitiveList * list, decaf::io::DataOutputStream & dataOut ) throw ( decaf::lang::Exception ) [static]`

Marshal a PrimitiveList to the given DataOutputStream.

#### Parameters

<i>list</i>	The list object to Marshal
<i>dataOut</i>	Reference to a DataOutputStream to write the marshaled data to.

#### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---



## 6.620 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference 2963

6.620.3.4 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalMap ( const util::PrimitiveMap \* *map*, decaf::io::DataOutputStream & *dataOut* ) throw ( decaf::lang::Exception ) [static]

Marshal a primitive map object to the given DataOutputStream.

### Parameters

<i>map</i>	Map to Marshal.
<i>dataOut</i>	Reference to a DataOutputStream to write the marshaled data to.

### Exceptions

<i>Exception</i>	if an error occurs during the marshaling process.
------------------	---

6.620.3.5 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive ( decaf::io::DataOutputStream & *dataOut*, const util::PrimitiveValueNode & *value* ) throw ( decaf::io::IOException ) [static, protected]

Used to Marshal the Primitive types out on the Wire.

### Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>value</i>	- the ValueNode to write.

### Exceptions

<i>IOException</i>	
--------------------	--

6.620.3.6 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList ( decaf::io::DataOutputStream & *dataOut*, const decaf::util::List< util::PrimitiveValueNode > & *list* ) throw ( decaf::io::IOException ) [static, protected]

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

### Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>list</i>	- the ValueNode to write.

### Exceptions

<i>IOException</i>	
--------------------	--

```
6.620.3.7 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap
( decaf::io::DataOutputStream & dataOut, const decaf::util::Map<
std::string, util::PrimitiveValueNode > & map ) throw (
decaf::io::IOException ) [static, protected]
```

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

#### Parameters

<i>dataOut</i>	- the DataOutputStream to write to
<i>map</i>	- the ValueNode to write.

#### Exceptions

<i>IOException</i>	
--------------------	--

```
6.620.3.8 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal
( util::PrimitiveList * list, const std::vector< unsigned char > & buffer ) throw (
decaf::lang::Exception ) [static]
```

Unmarshal a PrimitiveList from the provided byte buffer.

#### Parameters

<i>map</i>	The List to populate with values from the marshaled data.
<i>buffer</i>	The byte buffer containing the marshaled Map.

#### Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

```
6.620.3.9 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal
( util::PrimitiveMap * map, const std::vector< unsigned char > & buffer ) throw (
decaf::lang::Exception ) [static]
```

Unmarshal a PrimitiveMap from the provided Byte buffer.

#### Parameters

<i>map</i>	The Map to populate with values from the marshaled data.
<i>buffer</i>	The byte buffer containing the marshaled Map.

#### Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

## 6.620 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference 2965

6.620.3.10 `static util::PrimitiveList* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalList ( decaf::io::DataInputStream & dataIn ) throw ( decaf::lang::Exception )`  
[static]

Unmarshal a PrimitiveList from the given DataInputStream.

### Parameters

<i>dataIn</i>	The DataInputStream instance to read the marshaled PrimitiveList from.
---------------	--

### Returns

a pointer to a newly allocated PrimitiveList instance.

### Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

6.620.3.11 `static util::PrimitiveMap* activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalMap ( decaf::io::DataInputStream & dataIn ) throw ( decaf::lang::Exception )`  
[static]

Unmarshal a PrimitiveMap from the provided DataInputStream.

### Parameters

<i>dataIn</i>	The DataInputStream instance to read the marshaled PrimitiveMap from.
---------------	---

### Returns

a pointer to a newly allocated PrimitiveMap instance.

### Exceptions

<i>Exception</i>	if an error occurs during the unmarshal process.
------------------	--

6.620.3.12 `static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive ( decaf::io::DataInputStream & dataIn ) throw ( decaf::io::IOException )`  
[static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

### Parameters

<i>dataIn</i>	- DataInputStream to read from.
---------------	---------------------------------

### Returns

a PrimitiveValueNode containing the data.

**Exceptions**

<i>IOException</i>	
--------------------	--

**6.620.3.13** `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveList ( decaf::io::DataInputStream & dataIn, decaf::util::StlList< util::PrimitiveValueNode > & list ) throw ( decaf::io::IOException )` `[static, protected]`

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

**Parameters**

<i>dataIn</i>	- DataInputStream to read from.
<i>list</i>	- the ValueNode to write.

**Exceptions**

<i>IOException</i>	
--------------------	--

**6.620.3.14** `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveMap ( decaf::io::DataInputStream & dataIn, util::PrimitiveMap & map ) throw ( decaf::io::IOException )` `[static, protected]`

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

**Parameters**

<i>dataIn</i>	- DataInputStream to read from.
<i>map</i>	- the map to fill with data.

**Exceptions**

<i>IOException</i>	
--------------------	--

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**PrimitiveTypesMarshaller.h**

## 6.621 **activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference**

Define a union type comprised of the various types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

## Data Fields

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**
- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string \* **stringValue**
- std::vector< unsigned char > \* **byteArrayValue**
- decaf::util::List< PrimitiveValueNode > \* **listValue**
- decaf::util::Map< std::string, PrimitiveValueNode > \* **mapValue**

### 6.621.1 Detailed Description

Define a union type comprised of the various types.

### 6.621.2 Field Documentation

- 6.621.2.1 bool **activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue**
- 6.621.2.2 std::vector<unsigned char>\* **activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue**
- 6.621.2.3 unsigned char **activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue**
- 6.621.2.4 char **activemq::util::PrimitiveValueNode::PrimitiveValue::charValue**
- 6.621.2.5 double **activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue**
- 6.621.2.6 float **activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue**
- 6.621.2.7 int **activemq::util::PrimitiveValueNode::PrimitiveValue::intValue**
- 6.621.2.8 decaf::util::List<PrimitiveValueNode>\* **activemq::util::PrimitiveValueNode::PrimitiveValue::listValue**
- 6.621.2.9 long long **activemq::util::PrimitiveValueNode::PrimitiveValue::longValue**
- 6.621.2.10 decaf::util::Map<std::string, PrimitiveValueNode>\* **activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue**
- 6.621.2.11 short **activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue**

### 6.621.2.12 `std::string* activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue`

The documentation for this union was generated from the following file:

- `src/main/activemq/util/PrimitiveValueNode.h`

## 6.622 `activemq::util::PrimitiveValueConverter` Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2960) from one type to another.

```
#include <src/main/activemq/util/PrimitiveValueConverter.h>
```

### Public Member Functions

- **PrimitiveValueConverter** ()
- virtual **~PrimitiveValueConverter** ()
- `template<typename TO >`  
**TO convert** (const **PrimitiveValueNode** &value) const throw ( `decaf::lang::exceptions::UnsupportedOperat`  
 )

### 6.622.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2960) from one type to another.

If the conversion is supported then calling the convert method will throw an `UnsupportedOperationException` to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

			boolean	byte	short	int	long	float	double	String	-----														
-----		boolean		X X		byte		X X X X X		short		X X X X		int		X X X		long		X X		float		X X	
X		double		X X		String		X X X X X X X X		-----															

### Since

3.0

### 6.622.2 Constructor & Destructor Documentation

6.622.2.1 `activemq::util::PrimitiveValueConverter::PrimitiveValueConverter ( )` [`inline`]

6.622.2.2 `virtual activemq::util::PrimitiveValueConverter::~~PrimitiveValueConverter ( )`  
`[inline, virtual]`

### 6.622.3 Member Function Documentation

6.622.3.1 `std::vector< unsigned char > activemq::util::PrimitiveValueConverter::convert  
( const PrimitiveValueNode & value ) const throw (  
decaf::lang::exceptions::UnsupportedOperationException )  
[inline]`

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveValueConverter.h`

## 6.623 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

### Data Structures

- union **PrimitiveValue**  
*Define a union type comprised of the various types.*

### Public Types

- enum **PrimitiveType** {  
    **NULL\_TYPE** = 0, **BOOLEAN\_TYPE** = 1, **BYTE\_TYPE** = 2, **CHAR\_TYPE** = 3,  
    **SHORT\_TYPE** = 4, **INTEGER\_TYPE** = 5, **LONG\_TYPE** = 6, **DOUBLE\_TYPE** =  
    7,  
    **FLOAT\_TYPE** = 8, **STRING\_TYPE** = 9, **BYTE\_ARRAY\_TYPE** = 10, **MAP\_TYPE**  
    = 11,  
    **LIST\_TYPE** = 12, **BIG\_STRING\_TYPE** = 13 }  
*Enumeration for the various primitive types.*

### Public Member Functions

- **PrimitiveValueNode** ()  
*Default Constructor, creates a value of the NULL\_TYPE.*
- **PrimitiveValueNode** (bool value)  
*Boolean Value Constructor.*
- **PrimitiveValueNode** (unsigned char value)  
*Byte Value Constructor.*
- **PrimitiveValueNode** (char value)  
*Char Value Constructor.*

- **PrimitiveValueNode** (short value)  
*Short Value Constructor.*
- **PrimitiveValueNode** (int value)  
*Int Value Constructor.*
- **PrimitiveValueNode** (long long value)  
*Long Value Constructor.*
- **PrimitiveValueNode** (float value)  
*Float Value Constructor.*
- **PrimitiveValueNode** (double value)  
*Double Value Constructor.*
- **PrimitiveValueNode** (const char \*value)  
*String Value Constructor.*
- **PrimitiveValueNode** (const std::string &value)  
*String Value Constructor.*
- **PrimitiveValueNode** (const std::vector< unsigned char > &value)  
*Byte Array Value Constructor.*
- **PrimitiveValueNode** (const **decaf::util::List**< **PrimitiveValueNode** > &value)  
  
*Primitive List Constructor.*
- **PrimitiveValueNode** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &value)  
*Primitive Map Value Constructor.*
- **PrimitiveValueNode** (const **PrimitiveValueNode** &node)  
*Copy constructor.*
- **~PrimitiveValueNode** ()
- **PrimitiveValueNode & operator=** (const **PrimitiveValueNode** &node)  
*Assignment operator, copies the data from the other node.*
- bool **operator==** (const **PrimitiveValueNode** &node) const  
*Comparison Operator, compares this node to the other node.*
- **PrimitiveType getType** () const  
*Gets the Value Type of this type wrapper.*
- **PrimitiveValue getValue** () const  
*Gets the internal Primitive Value object from this wrapper.*
- void **setValue** (const **PrimitiveValue** &value, **PrimitiveType** valueType)  
*Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.*
- void **clear** ()  
*Clears the value from this wrapper converting it back to a blank NULL\_TYPE value.*
- void **setBool** (bool value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- bool **getBool** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
  
*Gets the Boolean value of this Node.*



- void **setByte** (unsigned char value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- unsigned char **getByte** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Byte value of this Node.*
- void **setChar** (char value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- char **getChar** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Character value of this Node.*
- void **setShort** (short value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- short **getShort** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Short value of this Node.*
- void **setInt** (int value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- int **getInt** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Integer value of this Node.*
- void **setLong** (long long value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- long long **getLong** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Long value of this Node.*
- void **setFloat** (float value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- float **getFloat** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Float value of this Node.*
- void **setDouble** (double value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- double **getDouble** () const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets the Double value of this Node.*
- void **setString** (const std::string &value)  
*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*
- std::string **getString** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the String value of this Node.*

- void **setByteArray** (const std::vector< unsigned char > &value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- std::vector< unsigned char > **getByteArray** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Byte Array value of this Node.*

- void **setList** (const decaf::util::List< PrimitiveValueNode > &value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- const decaf::util::List< PrimitiveValueNode > & **getList** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Primitive List value of this Node.*

- void **setMap** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)

*Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.*

- const decaf::util::Map< std::string, PrimitiveValueNode > & **getMap** () const throw ( decaf::lang::exceptions::NoSuchElementException )

*Gets the Primitive Map value of this Node.*

- std::string **toString** () const

*Creates a string representation of this value.*

### 6.623.1 Detailed Description

Class that wraps around a single value of one of the many types.

Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

### 6.623.2 Member Enumeration Documentation

#### 6.623.2.1 enum activemq::util::PrimitiveValueNode::PrimitiveType

Enumeration for the various primitive types.

Enumerator:

**NULL\_TYPE**

**BOOLEAN\_TYPE**

**BYTE\_TYPE**

**CHAR\_TYPE**

**SHORT\_TYPE**

*INTEGER\_TYPE*  
*LONG\_TYPE*  
*DOUBLE\_TYPE*  
*FLOAT\_TYPE*  
*STRING\_TYPE*  
*BYTE\_ARRAY\_TYPE*  
*MAP\_TYPE*  
*LIST\_TYPE*  
*BIG\_STRING\_TYPE*

### 6.623.3 Constructor & Destructor Documentation

#### 6.623.3.1 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( )`

Default Constructor, creates a value of the `NULL_TYPE`.

#### 6.623.3.2 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( bool value )`

Boolean Value Constructor.

##### Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

#### 6.623.3.3 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( unsigned char value )`

Byte Value Constructor.

##### Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

#### 6.623.3.4 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( char value )`

Char Value Constructor.

##### Parameters

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.5    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( short value )`**

Short Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.6    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( int value )`**

Int Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.7    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( long long value )`**

Long Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.8    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( float value )`**

Float Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.9    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( double value )`**

Double Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

**6.623.3.10    `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const char * value )`**

String Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

6.623.3.11 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const std::string & value )`

String Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

6.623.3.12 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const std::vector< unsigned char > & value )`

Byte Array Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

6.623.3.13 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const decaf::util::List< PrimitiveValueNode > & value )`

Primitive List Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

6.623.3.14 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const decaf::util::Map< std::string, PrimitiveValueNode > & value )`

Primitive Map Value Constructor.

**Parameters**

<i>value</i>	- the new value to store.
--------------	---------------------------

6.623.3.15 `activemq::util::PrimitiveValueNode::PrimitiveValueNode ( const PrimitiveValueNode & node )`

Copy constructor.

**Parameters**

<i>node</i>	The instance of another node to copy to this one.
-------------	---

6.623.3.16 `activemq::util::PrimitiveValueNode::~~PrimitiveValueNode ( ) [inline]`

**6.623.4 Member Function Documentation**

6.623.4.1 `void activemq::util::PrimitiveValueNode::clear ( )`

Clears the value from this wrapper converting it back to a blank NULL\_TYPE value.

6.623.4.2 `bool activemq::util::PrimitiveValueNode::getBool ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Boolean value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.3 `unsigned char activemq::util::PrimitiveValueNode::getByte ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Byte value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.4 `std::vector<unsigned char> activemq::util::PrimitiveValueNode::getByteArray ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Byte Array value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.5 `char activemq::util::PrimitiveValueNode::getChar ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Character value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.6 `double activemq::util::PrimitiveValueNode::getDouble ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Double value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.7 `float activemq::util::PrimitiveValueNode::getFloat ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Float value of this Node.

**Returns**

value contained at the given index

**Exceptions**

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.8 `int activemq::util::PrimitiveValueNode::getInt ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Integer value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.9 `const decaf::util::List<PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getList ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Primitive List value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.10 `long long activemq::util::PrimitiveValueNode::getLong ( ) const throw ( decaf::lang::exceptions::NoSuchElementException )`

Gets the Long value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---



6.623.4.11 `const decaf::util::Map<std::string, PrimitiveValueNode>&  
activemq::util::PrimitiveValueNode::getMap ( ) const throw (  
decaf::lang::exceptions::NoSuchElementException )`

Gets the Primitive Map value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.12 `short activemq::util::PrimitiveValueNode::getShort ( ) const throw (  
decaf::lang::exceptions::NoSuchElementException )`

Gets the Short value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.13 `std::string activemq::util::PrimitiveValueNode::getString ( ) const throw (  
decaf::lang::exceptions::NoSuchElementException )`

Gets the String value of this Node.

#### Returns

value contained at the given index

#### Exceptions

<i>NoSuchElementException</i>	this node cannot be returned as the requested type.
-------------------------------	---

6.623.4.14 `PrimitiveType activemq::util::PrimitiveValueNode::getType ( ) const  
[inline]`

Gets the Value Type of this type wrapper.

**Returns**

the `PrimitiveType` value for this wrapper.

**6.623.4.15** `PrimitiveValue` `activemq::util::PrimitiveValueNode::getValue ( ) const`  
`[inline]`

Gets the internal Primitive Value object from this wrapper.

**Returns**

a copy of the contained **PrimitiveValue** (p. 2957)

**6.623.4.16** `PrimitiveValueNode&` `activemq::util::PrimitiveValueNode::operator= ( const`  
`PrimitiveValueNode & node )`

Assignment operator, copies the data from the other node.

**Parameters**

<i>node</i>	The instance of another node to copy to this one.
-------------	---

**6.623.4.17** `bool` `activemq::util::PrimitiveValueNode::operator== ( const PrimitiveValueNode`  
`& node ) const`

Comparison Operator, compares this node to the other node.

**Returns**

true if the values are the same false otherwise.

**6.623.4.18** `void` `activemq::util::PrimitiveValueNode::setBool ( bool value )`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.19** `void` `activemq::util::PrimitiveValueNode::setByte ( unsigned char value )`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.20** void activemq::util::PrimitiveValueNode::setByteArray ( const std::vector< unsigned char > & *value* )

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.21** void activemq::util::PrimitiveValueNode::setChar ( char *value* )

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.22** void activemq::util::PrimitiveValueNode::setDouble ( double *value* )

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

**6.623.4.23** void activemq::util::PrimitiveValueNode::setFloat ( float *value* )

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

**Parameters**

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.623.4.24 `void activemq::util::PrimitiveValueNode::setInt ( int value )`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

#### Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.623.4.25 `void activemq::util::PrimitiveValueNode::setList ( const decaf::util::List< PrimitiveValueNode > & value )`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

#### Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.623.4.26 `void activemq::util::PrimitiveValueNode::setLong ( long long value )`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

#### Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.623.4.27 `void activemq::util::PrimitiveValueNode::setMap ( const decaf::util::Map< std::string, PrimitiveValueNode > & value )`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

#### Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.623.4.28 `void activemq::util::PrimitiveValueNode::setShort ( short value )`

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

#### Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.623.4.29 void activemq::util::PrimitiveValueNode::setString ( const std::string & *value* )

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

#### Parameters

<i>value</i>	- the new value to assign to the element at index
--------------	---

6.623.4.30 void activemq::util::PrimitiveValueNode::setValue ( const PrimitiveValue & *value*, PrimitiveType *valueType* )

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

#### Parameters

<i>value</i>	The value to set as the value contained in this Node.
<i>valueType</i>	The type of the value being set into this one.

6.623.4.31 std::string activemq::util::PrimitiveValueNode::toString ( ) const

Creates a string representation of this value.

#### Returns

string value of this type wrapper.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

## 6.624 decaf::security::Principal Class Reference

Base interface for a principal, which can represent an individual or organization.

```
#include <src/main/decaf/security/Principal.h>
```

Inheritance diagram for decaf::security::Principal:

#### Public Member Functions

- virtual ~**Principal** ()
- virtual bool **equals** (const **Principal** &another) const =0

*Compares two principals to see if they are the same.*

- `virtual std::string getName () const =0`

*Provides the name of this principal.*

### 6.624.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

### 6.624.2 Constructor & Destructor Documentation

6.624.2.1 `virtual decaf::security::Principal::~~Principal ( ) [inline, virtual]`

### 6.624.3 Member Function Documentation

6.624.3.1 `virtual bool decaf::security::Principal::equals ( const Principal & another ) const [pure virtual]`

Compares two principals to see if they are the same.

#### Parameters

<i>another</i>	A principal to be tested for equality to this one.
----------------	--

#### Returns

true if the given principal is equivalent to this one.

6.624.3.2 `virtual std::string decaf::security::Principal::getName ( ) const [pure virtual]`

Provides the name of this principal.

#### Returns

the name of this principal.

Implemented in `decaf::security::auth::x500::X500Principal` (p. 3957).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/Principal.h`

## 6.625 decaf::util::PriorityQueue< E > Class Template Reference

An unbounded priority queue based on a binary heap algorithm.

```
#include <src/main/decaf/util/PriorityQueue.h>
```

Inheritance diagram for decaf::util::PriorityQueue< E >:

## Data Structures

- class **ConstPriorityQueueIterator**
- class **PriorityQueueIterator**

## Public Member Functions

- **PriorityQueue** ()  
*Creates a Priority **Queue** (p. 3094) with the default initial capacity.*
- **PriorityQueue** (std::size\_t initialCapacity)  
*Creates a Priority **Queue** (p. 3094) with the capacity value supplied.*
- **PriorityQueue** (std::size\_t initialCapacity, **Comparator**< E > \*comparator)  
*Creates a Priority **Queue** (p. 3094) with the default initial capacity.*
- **PriorityQueue** (const **Collection**< E > &source)  
*Creates a **PriorityQueue** (p. 2975) containing the elements in the specified **Collection** (p. 1155).*
- **PriorityQueue** (const **PriorityQueue**< E > &source)  
*Creates a **PriorityQueue** (p. 2975) containing the elements in the specified priority queue.*
- virtual ~**PriorityQueue** ()
- **PriorityQueue**< E > & **operator=** (const **Collection**< E > &source)  
*Assignment operator, assign another **Collection** (p. 1155) to this one.*
- **PriorityQueue**< E > & **operator=** (const **PriorityQueue**< E > &source)  
*Assignment operator, assign another **PriorityQueue** (p. 2975) to this one.*
- virtual **decaf::util::Iterator**< E > \* **iterator** ()
- virtual **decaf::util::Iterator**< E > \* **iterator** () const
- virtual std::size\_t **size** () const  
*Returns the number of elements in this collection.*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*
- virtual bool **offer** (const E &value) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the queue provided that the condition allows such an operation.*
- virtual bool **poll** (E &result)  
*Gets and removes the element in the head of the queue.*
- virtual bool **peek** (E &result) const  
*Gets but not removes the element in the head of the queue.*
- virtual E **remove** () throw ( decaf::lang::exceptions::NoSuchElementException )

*Retrieves and removes the head of this queue.*

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

*Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.*

- **decaf::lang::Pointer< Comparator< E > > comparator** () const

*obtains a Copy of the Pointer instance that this **PriorityQueue** (p.2975) is using to compare the elements in the queue with.*

## Friends

- class **PriorityQueueIterator**

### 6.625.1 Detailed Description

```
template<typename E>class decaf::util::PriorityQueue< E >
```

An unbounded priority queue based on a binary heap algorithm.

The elements of the priority queue are ordered according to their natural ordering, or by a **Comparator** (p.1189) provided to one of the constructors that accepts Comparators. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in a compiler error).

The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

A priority queue is unbounded, but has an internal capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the optional methods of the **Collection** (p.1155) and **Iterator** (p.2114) interfaces. The **Iterator** (p.2114) provided in method **iterator()** (p.2980) is not guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using `Arrays::sort ( pq.toArray() )`.

Note that this implementation is not synchronized. Multiple threads should not access a **PriorityQueue** (p.2975) instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe **PriorityBlockingQueue** class.



Implementation note: this implementation provides  $O(\log(n))$  time for the enqueueing and dequeuing methods (offer, poll, **remove()** (p. 2982) and add); linear time for the remove(Object) and contains(Object) methods; and constant time for the retrieval methods (peek, element, and size).

### Since

1.0

## 6.625.2 Constructor & Destructor Documentation

6.625.2.1 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( )`  
[inline]

Creates a **Priority Queue** (p. 3094) with the default initial capacity.

6.625.2.2 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (`  
`std::size_t initialCapacity )` [inline]

Creates a **Priority Queue** (p. 3094) with the capacity value supplied.

### Parameters

<i>initialCapacity</i>	The initial number of elements allocated to this <b>PriorityQueue</b> (p. 2975).
------------------------	--

6.625.2.3 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (`  
`std::size_t initialCapacity, Comparator< E > * comparator )` [inline]

Creates a **Priority Queue** (p. 3094) with the default initial capacity.

This new **PriorityQueue** (p. 2975) takes ownership of the passed **Comparator** (p. 1189) instance and uses that to determine the ordering of the elements in the **Queue** (p. 3094).

### Parameters

<i>initialCapacity</i>	The initial number of elements allocated to this <b>PriorityQueue</b> (p. 2975).
<i>comparator</i>	The <b>Comparator</b> (p. 1189) instance to use in sorting the elements in the <b>Queue</b> (p. 3094).

### Exceptions

<i>NullPointerException</i>	if the passed <b>Comparator</b> (p. 1189) is NULL.
-----------------------------	--

6.625.2.4 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( const Collection< E > & source ) [inline]`

Creates a **PriorityQueue** (p. 2975) containing the elements in the specified **Collection** (p. 1155).

#### Parameters

<i>source</i>	the <b>Collection</b> (p. 1155) whose elements are to be placed into this priority queue
---------------	--

6.625.2.5 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue ( const PriorityQueue< E > & source ) [inline]`

Creates a **PriorityQueue** (p. 2975) containing the elements in the specified priority queue.

This priority queue will be ordered according to the same ordering as the given priority queue.

#### Parameters

<i>source</i>	the priority queue whose elements are to be placed into this priority queue
---------------	---

6.625.2.6 `template<typename E> virtual decaf::util::PriorityQueue< E >::~~PriorityQueue ( ) [inline, virtual]`

### 6.625.3 Member Function Documentation

6.625.3.1 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::add ( const E & value ) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException ) [inline, virtual]`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

#### Parameters

<i>value</i>	- the element to offer to the <b>Queue</b> (p. 3094).
--------------	---

#### Returns

true if the add succeeds.

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the element cannot be added.
---------------------------------------	---------------------------------

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 165).

References DECAF\_CATCH\_EXCEPTION\_CONVERT, DECAF\_CATCH\_RETHROW, DECAF\_CATCHALL\_THROW, and decaf::util::PriorityQueue< E >::offer().

```
6.625.3.2  template<typename E> virtual void decaf::util::PriorityQueue< E >::clear
           ( ) throw ( lang::exceptions::UnsupportedOperationException )
           [inline, virtual]
```

Removes all elements of the queue.

This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 166).

```
6.625.3.3  template<typename E> decaf::lang::Pointer< Comparator<E> >
           decaf::util::PriorityQueue< E >::comparator ( ) const [inline]
```

obtains a Copy of the Pointer instance that this **PriorityQueue** (p.2975) is using to compare the elements in the queue with.

The returned value is a copy, the caller cannot change the value if the internal Pointer value.

**Returns**

a copy of the **Comparator** (p. 1189) Pointer being used by this **Queue** (p. 3094).

```
6.625.3.4  template<typename E> virtual decaf::util::Iterator<E>*
           decaf::util::PriorityQueue< E >::iterator ( ) const [inline,
           virtual]
```

Implements **decaf::lang::Iterable< E >** (p.2114).

```
6.625.3.5  template<typename E> virtual decaf::util::Iterator<E>*
           decaf::util::PriorityQueue< E >::iterator ( ) [inline, virtual]
```

**Returns**

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p.2113).

References decaf::util::PriorityQueue< E >::PriorityQueueIterator.

```
6.625.3.6  template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer (
            const E & value ) throw ( decaf::lang::exceptions::NullPointerException,
            decaf::lang::exceptions::IllegalArgumentException ) [inline,
            virtual]
```

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

#### Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

#### Returns

true if the operation succeeds and false if it fails.

#### Exceptions

<i>NullPointerException</i>	if the <b>Queue</b> (p. 3094) implementation does not allow Null values to be inserted into the <b>Queue</b> (p. 3094).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 3096).

Referenced by `decaf::util::PriorityQueue< E >::add()`.

```
6.625.3.7  template<typename E> PriorityQueue<E> & decaf::util::PriorityQueue< E
            >::operator= ( const PriorityQueue< E > & source ) [inline]
```

Assignment operator, assign another **PriorityQueue** (p. 2975) to this one.

#### Parameters

<i>source</i>	The <b>PriorityQueue</b> (p. 2975) to copy to this one.
---------------	---

```
6.625.3.8  template<typename E> PriorityQueue<E> & decaf::util::PriorityQueue< E
            >::operator= ( const Collection< E > & source ) [inline]
```

Assignment operator, assign another **Collection** (p. 1155) to this one.

#### Parameters

<i>source</i>	The <b>Collection</b> (p. 1155) to copy to this one.
---------------	--

6.625.3.9 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::peek ( E  
& result ) const [inline, virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

#### Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

#### Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 3097).

6.625.3.10 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::poll ( E  
& result ) [inline, virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 3094) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

#### Parameters

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

#### Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 3097).

6.625.3.11 `template<typename E> virtual E decaf::util::PriorityQueue< E >::remove  
( ) throw ( decaf::lang::exceptions::NoSuchElementException )  
[inline, virtual]`

Retrieves and removes the head of this queue.

This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

#### Returns

a copy of the element in the head of the queue.

**Exceptions**

<i>NoSuchElementException</i>	if the queue is empty.
-------------------------------	------------------------

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 167).

```
6.625.3.12  template<typename E> virtual bool decaf::util::PriorityQueue<
              E >::remove ( const E & value ) throw (
              lang::exceptions::UnsupportedOperationException,
              lang::exceptions::IllegalArgumentException ) [inline,
              virtual]
```

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

**Parameters**

<i>value</i>	- element to be removed from this collection, if present
--------------	--

**Returns**

true if an element was removed as a result of this call

**Exceptions**

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1155).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 156).

```
6.625.3.13  template<typename E> virtual std::size_t decaf::util::PriorityQueue< E
              >::size ( ) const [inline, virtual]
```

Returns the number of elements in this collection.

If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

**Returns**

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 1164).

**6.625.4 Friends And Related Function Documentation**

6.625.4.1 `template<typename E> friend class PriorityQueueIterator` [*friend*]

Referenced by `decaf::util::PriorityQueue`< **E** >::iterator().

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ProducerAck.h`

**6.626 activemq::commands::ProducerAck Class Reference**

```
#include <src/main/activemq/commands/ProducerAck.h>
```

Inheritance diagram for `activemq::commands::ProducerAck`:

**Public Member Functions**

- **ProducerAck** ()
- virtual **~ProducerAck** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerAck \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual int **getSize** () const

- virtual void **setSize** (int **size**)
- virtual bool **isProducerAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_PRODUCERACK** = 19

### Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- int **size**

## 6.626.1 Constructor & Destructor Documentation

6.626.1.1 **activemq::commands::ProducerAck::ProducerAck** ( )

6.626.1.2 **virtual activemq::commands::ProducerAck::~~ProducerAck** ( ) [virtual]

## 6.626.2 Member Function Documentation

6.626.2.1 **virtual ProducerAck\*** **activemq::commands::ProducerAck::cloneDataStructure** ( )  
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.626.2.2 **virtual void** **activemq::commands::ProducerAck::copyDataStructure** ( const  
**DataStructure** \* **src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------



Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.626.2.3 `virtual bool activemq::commands::ProducerAck::equals ( const DataStructure *  
value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.626.2.4 `virtual unsigned char activemq::commands::ProducerAck::getDataStructureType ( )  
const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.626.2.5 `virtual const Pointer<ProducerId>& ac-  
tivemq::commands::ProducerAck::getProducerId ( ) const  
[virtual]`

6.626.2.6 `virtual Pointer<ProducerId>& ac-  
tivemq::commands::ProducerAck::getProducerId ( )  
[virtual]`

6.626.2.7 `virtual int activemq::commands::ProducerAck::getSize ( ) const [virtual]`

6.626.2.8 `virtual bool activemq::commands::ProducerAck::isProducerAck ( ) const  
[inline, virtual]`

#### Returns

an answer of true to the **isProducerAck()** (p. 2986) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 727).

6.626.2.9 `virtual void activemq::commands::ProducerAck::setProducerId ( const Pointer< ProducerId > & producerId ) [virtual]`

6.626.2.10 `virtual void activemq::commands::ProducerAck::setSize ( int size ) [virtual]`

6.626.2.11 `virtual std::string activemq::commands::ProducerAck::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.626.2.12 `virtual Pointer<Command> activemq::commands::ProducerAck::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.626.3 Field Documentation

6.626.3.1 `const unsigned char activemq::commands::ProducerAck::ID_- PRODUCERACK = 19 [static]`

6.626.3.2 `Pointer<ProducerId> activemq::commands::ProducerAck::producerId [protected]`

6.626.3.3 `int activemq::commands::ProducerAck::size [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerAck.h`

## 6.627 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2988).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.627.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2988).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.627.2 Constructor & Destructor Documentation

6.627.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`

6.627.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.627.3 Member Function Documentation

6.627.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.627.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.627.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.627.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

6.627.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.627.3.6  virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.627.3.7  virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerAckMarshaller.h**

## 6.628 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2992).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.628.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2992).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.628.2 Constructor & Destructor Documentation

6.628.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`

6.628.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.628.3 Member Function Documentation

6.628.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.628.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.628.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.628.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

6.628.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.628.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.628.3.7  virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerAckMarshaller.h**

## 6.629 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2996).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.629.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2996).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.629.2 Constructor & Destructor Documentation

6.629.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`

6.629.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.629.3 Member Function Documentation

6.629.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.629.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.629.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.629.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.629.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.629.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.629.3.7  virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerAckMarshaller.h**

## 6.630 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3000).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.630.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3000).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.630.2 Constructor & Destructor Documentation

6.630.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`

6.630.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.630.3 Member Function Documentation

6.630.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.630.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.630.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.630.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

6.630.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.630.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.630.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerAckMarshaller.h**

## 6.631 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3004).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller:

### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.631.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3004).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.631.2 Constructor & Destructor Documentation

6.631.2.1 `activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`

6.631.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.631.3 Member Function Documentation

6.631.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.631.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.631.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.631.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.631.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.631.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.631.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ProducerAckMarshaller.h**

## 6.632 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3008).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller:

#### Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

#### 6.632.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3008).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.632.2 Constructor & Destructor Documentation

6.632.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::ProducerAckMarshaller ( ) [inline]`

6.632.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::~~ProducerAckMarshaller ( ) [inline, virtual]`

## 6.632.3 Member Function Documentation

6.632.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.632.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.632.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.632.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

6.632.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.632.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.632.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerAckMarshaller.h**

## 6.633 activemq::cmsutil::ProducerCallback Class Reference

Callback for sending a message to a CMS destination.

```
#include <src/main/activemq/cmsutil/ProducerCallback.h>
```

Inheritance diagram for activemq::cmsutil::ProducerCallback:

### Public Member Functions

- virtual **~ProducerCallback** ()
- virtual void **dolnCms** (**cms::Session** \*session, **cms::MessageProducer** \*producer)=0  
throw ( cms::CMSException )

*Execute an action given a session and producer.*

#### 6.633.1 Detailed Description

Callback for sending a message to a CMS destination.

#### 6.633.2 Constructor & Destructor Documentation

6.633.2.1 virtual activemq::cmsutil::ProducerCallback::~~ProducerCallback ( ) [inline, virtual]

#### 6.633.3 Member Function Documentation

6.633.3.1 virtual void activemq::cmsutil::ProducerCallback::dolnCms ( **cms::Session** \* session, **cms::MessageProducer** \* producer ) throw ( cms::CMSException )  
[pure virtual]

Execute an action given a session and producer.

#### Parameters

<i>session</i>	the CMS Session
<i>producer</i>	the CMS Producer

#### Exceptions

<b>cms::CMSException</b> (p. 1130)	if thrown by CMS API methods
---------------------------------------	------------------------------

Implemented in **activemq::cmsutil::CmsTemplate::SendExecutor** (p. 3291).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ProducerCallback.h**

## 6.634 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ProducerExecutor:

### Public Member Functions

- **ProducerExecutor** (**ProducerCallback** \*action, **CmsTemplate** \*parent, **cms::Destination** \*destination)
- virtual ~**ProducerExecutor** ()
- virtual void **doInCms** (**cms::Session** \*session) throw ( **cms::CMSException** )  
*Execute any number of operations against the supplied CMS session.*
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \*session **AMQCPP\_UNUSED**) throw ( **cms::CMSException** )

### Protected Member Functions

- **ProducerExecutor** (const **ProducerExecutor** &)
- **ProducerExecutor** & **operator=** (const **ProducerExecutor** &)

### Protected Attributes

- **ProducerCallback** \* action
- **CmsTemplate** \* parent
- **cms::Destination** \* destination

### 6.634.1 Constructor & Destructor Documentation

6.634.1.1 **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** ( const **ProducerExecutor** & ) [*inline, protected*]

6.634.1.2 **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** ( **ProducerCallback** \* action, **CmsTemplate** \* parent, **cms::Destination** \* destination ) [*inline*]

6.634.1.3 **virtual activemq::cmsutil::CmsTemplate::ProducerExecutor::~~ProducerExecutor** ( ) [*inline, virtual*]

## 6.634.2 Member Function Documentation

6.634.2.1 `virtual void activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms ( cms::Session * session ) throw ( cms::CMSException ) [virtual]`

Execute any number of operations against the supplied CMS session.

### Parameters

<i>session</i>	the CMS Session
----------------	-----------------

### Exceptions

<b><i>cms::CMSException</i></b> (p. 1130)	if thrown by CMS API methods
--	------------------------------

Implements `activemq::cmsutil::SessionCallback` (p. 3320).

6.634.2.2 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination ( cms::Session *session AMQCPP_UNUSED ) throw ( cms::CMSException ) [inline, virtual]`

6.634.2.3 `ProducerExecutor& activemq::cmsutil::CmsTemplate::ProducerExecutor::operator= ( const ProducerExecutor & ) [inline, protected]`

## 6.634.3 Field Documentation

6.634.3.1 `ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action [protected]`

6.634.3.2 `cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination [protected]`

6.634.3.3 `CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.635 activemq::commands::ProducerId Class Reference

```
#include <src/main/activemq/commands/ProducerId.h>
```

Inheritance diagram for `activemq::commands::ProducerId`:

## Public Types

- typedef **decaf::lang::PointerComparator**< **ProducerId** > **COMPARATOR**

## Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &sessionId, long long consumerId)
- **ProducerId** (std::string producerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- void **setProducerSessionKey** (std::string sessionKey)
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual int **compareTo** (const **ProducerId** &value) const
- virtual bool **equals** (const **ProducerId** &value) const
- virtual bool **operator==** (const **ProducerId** &value) const
- virtual bool **operator<** (const **ProducerId** &value) const
- **ProducerId** & **operator=** (const **ProducerId** &other)

## Static Public Attributes

- static const unsigned char **ID\_PRODUCERID** = 123

### Protected Attributes

- `std::string` **connectionId**
- `long long` **value**
- `long long` **sessionId**

### 6.635.1 Member Typedef Documentation

6.635.1.1 `typedef decaf::lang::PointerComparator<ProducerId>`  
`activemq::commands::ProducerId::COMPARATOR`

### 6.635.2 Constructor & Destructor Documentation

6.635.2.1 `activemq::commands::ProducerId::ProducerId ( )`

6.635.2.2 `activemq::commands::ProducerId::ProducerId ( const ProducerId & other )`

6.635.2.3 `activemq::commands::ProducerId::ProducerId ( const SessionId & sessionId, long long consumerId )`

6.635.2.4 `activemq::commands::ProducerId::ProducerId ( std::string producerId )`

6.635.2.5 `virtual activemq::commands::ProducerId::~~ProducerId ( )` `[virtual]`

### 6.635.3 Member Function Documentation

6.635.3.1 `virtual ProducerId* activemq::commands::ProducerId::cloneDataStructure ( )`  
`const` `[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1628).

6.635.3.2 `virtual int activemq::commands::ProducerId::compareTo ( const ProducerId & value ) const` `[virtual]`

6.635.3.3 `virtual void activemq::commands::ProducerId::copyDataStructure ( const DataStructure * src )` `[virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.635.3.4 **virtual bool** **activemq::commands::ProducerId::equals** ( **const DataStructure \*** *value* ) **const** [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.635.3.5 **virtual bool** **activemq::commands::ProducerId::equals** ( **const ProducerId &** *value* ) **const** [virtual]

6.635.3.6 **virtual const std::string&** **activemq::commands::ProducerId::getConnectionId** ( ) **const** [virtual]

6.635.3.7 **virtual std::string&** **activemq::commands::ProducerId::getConnectionId** ( ) [virtual]

6.635.3.8 **virtual unsigned char** **activemq::commands::ProducerId::getDataStructureType** ( ) **const** [virtual]

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.635.3.9 **const Pointer<SessionId>&** **activemq::commands::ProducerId::getParentId** ( ) **const**

6.635.3.10 **virtual long long** **activemq::commands::ProducerId::getSessionId** ( ) **const** [virtual]

6.635.3.11 **virtual long long** **activemq::commands::ProducerId::getValue** ( ) **const** [virtual]



- 6.635.3.12 `virtual bool activemq::commands::ProducerId::operator< ( const ProducerId & value ) const` [virtual]
- 6.635.3.13 `ProducerId& activemq::commands::ProducerId::operator= ( const ProducerId & other )`
- 6.635.3.14 `virtual bool activemq::commands::ProducerId::operator== ( const ProducerId & value ) const` [virtual]
- 6.635.3.15 `virtual void activemq::commands::ProducerId::setConnectionId ( const std::string & connectionId )` [virtual]
- 6.635.3.16 `void activemq::commands::ProducerId::setProducerSessionKey ( std::string sessionKey )`
- 6.635.3.17 `virtual void activemq::commands::ProducerId::setSessionId ( long long sessionId )` [virtual]
- 6.635.3.18 `virtual void activemq::commands::ProducerId::setValue ( long long value )` [virtual]
- 6.635.3.19 `virtual std::string activemq::commands::ProducerId::toString ( ) const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 796).

#### 6.635.4 Field Documentation

- 6.635.4.1 `std::string activemq::commands::ProducerId::connectionId` [protected]
- 6.635.4.2 `const unsigned char activemq::commands::ProducerId::ID_PRODUCERID = 123` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.635.4.3 `long long activemq::commands::ProducerId::sessionId` [protected]
- 6.635.4.4 `long long activemq::commands::ProducerId::value` [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerId.h**

## 6.636 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3019).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarsh
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.636.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3019).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.636.2 Constructor & Destructor Documentation

6.636.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::ProducerIdMarshaller ( ) [inline]`

6.636.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::~~ProducerIdMarshaller ( ) [inline, virtual]`

### 6.636.3 Member Function Documentation

6.636.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.636.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.636.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.636.3.4  virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.636.3.5  virtual int activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.636.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.636.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h`

## 6.637 `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3023).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller`:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.637.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3023).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.637.2 Constructor & Destructor Documentation

6.637.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::ProducerIdMarshaller ( ) [inline]`

6.637.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::~~ProducerIdMarshaller ( ) [inline, virtual]`

### 6.637.3 Member Function Documentation

6.637.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.637.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.637.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.637.3.4  virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.637.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.637.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.637.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h`

## 6.638 `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3027).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarsh
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller`:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.638.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3027).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.638.2 Constructor & Destructor Documentation

**6.638.2.1** `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::ProducerIdMarshaller ( ) [inline]`

**6.638.2.2** `virtual activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::~~ProducerIdMarshaller ( ) [inline, virtual]`

### 6.638.3 Member Function Documentation

**6.638.3.1** `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

**6.638.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.638.3.3** `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.638.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.638.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.638.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.638.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerIdMarshaller.h**

## 6.639 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3031).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarsh
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.639.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3031).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.639.2 Constructor & Destructor Documentation

6.639.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::ProducerIdMarshaller ( ) [inline]`

6.639.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::~~ProducerIdMarshaller ( ) [inline, virtual]`

### 6.639.3 Member Function Documentation

6.639.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.639.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.639.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.639.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.639.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.



## 6.639 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference 3043

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.639.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.639.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerIdMarshaller.h**

## 6.640 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3035).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarsh
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.640.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3035).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.640.2 Constructor & Destructor Documentation

6.640.2.1 `activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::ProducerIdMarshaller ( ) [inline]`

6.640.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::~~ProducerIdMarshaller ( ) [inline, virtual]`

### 6.640.3 Member Function Documentation

6.640.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.640.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.640.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.640.3.4  virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.640.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.640.3.6 virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.640.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ProducerIdMarshaller.h**

## 6.641 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3039).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarsh
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller:

### Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.641.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3039).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.641.2 Constructor & Destructor Documentation

6.641.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::ProducerIdMarshaller  
( ) [inline]`

6.641.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::~~ProducerIdMarshaller  
( ) [inline, virtual]`

### 6.641.3 Member Function Documentation

6.641.3.1 `virtual commands::DataStructure* ac-  
tivemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::createObject ( )  
const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.641.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::getDataStructureType  
( )const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.641.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.641.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.641.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.641.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal2 ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataOutputStream** \* *dataOut*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.641.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshall/v1/**ProducerIdMarshaller.h**

## 6.642 activemq::commands::ProducerInfo Class Reference

```
#include <src/main/activemq/commands/ProducerInfo.h>
```

Inheritance diagram for activemq::commands::ProducerInfo:

### Public Member Functions

- **ProducerInfo** ()
- virtual **~ProducerInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ProducerInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int windowSize)

- virtual bool **isProducerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_PRODUCERINFO** = 6

### Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::vector< decaf::lang::Pointer< **BrokerId** > > **brokerPath**
- bool **dispatchAsync**
- int **windowSize**

## 6.642.1 Constructor & Destructor Documentation

6.642.1.1 **activemq::commands::ProducerInfo::ProducerInfo** ( )

6.642.1.2 **virtual activemq::commands::ProducerInfo::~~ProducerInfo** ( ) [virtual]

## 6.642.2 Member Function Documentation

6.642.2.1 **virtual ProducerInfo\*** **activemq::commands::ProducerInfo::cloneDataStructure** ( )  
**const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.642.2.2 **virtual void** **activemq::commands::ProducerInfo::copyDataStructure** ( **const DataStructure** \* *src* ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.642.2.3 **Pointer<RemoveInfo>** **activemq::commands::ProducerInfo::createRemoveCommand**  
( ) const

6.642.2.4 **virtual bool** **activemq::commands::ProducerInfo::equals** ( const **DataStructure** \*  
*value* ) const [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.642.2.5 **virtual const std::vector< decaf::lang::Pointer<BrokerId> >&**  
**activemq::commands::ProducerInfo::getBrokerPath** ( ) const [virtual]

6.642.2.6 **virtual std::vector< decaf::lang::Pointer<BrokerId> >&**  
**activemq::commands::ProducerInfo::getBrokerPath** ( ) [virtual]

6.642.2.7 **virtual unsigned char** **activemq::commands::ProducerInfo::getDataStructureType** ( )  
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.642.2.8 **virtual const Pointer<ActiveMQDestination>&**  
**activemq::commands::ProducerInfo::getDestination** ( ) const [virtual]

6.642.2.9 **virtual Pointer<ActiveMQDestination>&**  
**activemq::commands::ProducerInfo::getDestination** ( ) [virtual]

6.642.2.10 **virtual Pointer<ProducerId>&** **ac-**  
**tivemq::commands::ProducerInfo::getProducerId** ( )  
[virtual]

- 6.642.2.11 `virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId ( ) const`  
[virtual]
- 6.642.2.12 `virtual int activemq::commands::ProducerInfo::getWindowSize ( ) const`  
[virtual]
- 6.642.2.13 `virtual bool activemq::commands::ProducerInfo::isDispatchAsync ( ) const`  
[virtual]
- 6.642.2.14 `virtual bool activemq::commands::ProducerInfo::isProducerInfo ( ) const`  
[inline, virtual]

### Returns

an answer of true to the **isProducerInfo()** (p. 3046) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 727).

- 6.642.2.15 `virtual void activemq::commands::ProducerInfo::setBrokerPath ( const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath )` [virtual]
- 6.642.2.16 `virtual void activemq::commands::ProducerInfo::setDestination ( const Pointer< ActiveMQDestination > & destination )` [virtual]
- 6.642.2.17 `virtual void activemq::commands::ProducerInfo::setDispatchAsync ( bool dispatchAsync )` [virtual]
- 6.642.2.18 `virtual void activemq::commands::ProducerInfo::setProducerId ( const Pointer< ProducerId > & producerId )` [virtual]
- 6.642.2.19 `virtual void activemq::commands::ProducerInfo::setWindowSize ( int windowSize )`  
[virtual]
- 6.642.2.20 `virtual std::string activemq::commands::ProducerInfo::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

```
6.642.2.21 virtual Pointer<Command> activemq::commands::ProducerInfo::visit
( activemq::state::CommandVisitor * visitor ) throw (
exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

## 6.642.3 Field Documentation

```
6.642.3.1 std::vector< decaf::lang::Pointer<BrokerId> >
activemq::commands::ProducerInfo::brokerPath [protected]
```

```
6.642.3.2 Pointer<ActiveMQDestination> ac-
tivemq::commands::ProducerInfo::destination
[protected]
```

```
6.642.3.3 bool activemq::commands::ProducerInfo::dispatchAsync
[protected]
```

```
6.642.3.4 const unsigned char activemq::commands::ProducerInfo::ID_ -
PRODUCERINFO = 6 [static]
```

```
6.642.3.5 Pointer<ProducerId> activemq::commands::ProducerInfo::producerId
[protected]
```

```
6.642.3.6 int activemq::commands::ProducerInfo::windowSize [protected]
```

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerInfo.h**

## 6.643 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3047).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMar
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller:

## Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual ~**ProducerInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.643.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3047).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.643.2 Constructor & Destructor Documentation

6.643.2.1 **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::ProducerInfoMarshaller**  
( ) [inline]

6.643.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

### 6.643.3 Member Function Documentation

6.643.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.643.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.643.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).



6.643.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**  
 (p. 739).

6.643.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal1  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**  
 (p. 740).

```
6.643.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.643.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerInfoMarshaller.h**

## 6.644 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3052).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.644.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3052).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.644.2 Constructor & Destructor Documentation

6.644.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]`

6.644.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

## 6.644.3 Member Function Documentation

6.644.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.644.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.644.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.644.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.644.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.644.3.6  virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.644.3.7  virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerInfoMarshaller.h**

## 6.645 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3056).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.645.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3056).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.645.2 Constructor & Destructor Documentation

6.645.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]`

6.645.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

## 6.645.3 Member Function Documentation

6.645.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.645.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.645.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.645.3.4  virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.645.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.645.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.645.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerInfoMarshaller.h**

## 6.646 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3060).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.646.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3060).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.646.2 Constructor & Destructor Documentation

6.646.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]`

6.646.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

## 6.646.3 Member Function Documentation

6.646.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.646.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.646.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.646.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

6.646.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.646.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.646.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ProducerInfoMarshaller.h**

## 6.647 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3064).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.647.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3064).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.647.2 Constructor & Destructor Documentation

6.647.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]`

6.647.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

## 6.647.3 Member Function Documentation

6.647.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.647.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.647.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.647.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.647.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.647.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.647.3.7  virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerInfoMarshaller.h**

## 6.648 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3068).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller:

### Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.648.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3068).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.648.2 Constructor & Destructor Documentation

6.648.2.1 `activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::ProducerInfoMarshaller ( ) [inline]`

6.648.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::~~ProducerInfoMarshaller ( ) [inline, virtual]`

## 6.648.3 Member Function Documentation

6.648.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.648.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.648.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.648.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.648.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

## Returns

int value indicating the size of the marshaled object.

## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.648.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.648.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ProducerInfoMarshaller.h**

## 6.649 activemq::state::ProducerState Class Reference

```
#include <src/main/activemq/state/ProducerState.h>
```

### Public Member Functions

- **ProducerState** (const **Pointer**< **ProducerInfo** > &info)
- virtual ~**ProducerState** ()
- std::string **toString** () const
- const **Pointer**< **ProducerInfo** > & **getInfo** () const
- void **setTransactionState** (const **Pointer**< **TransactionState** > &transactionState)
- **Pointer**< **TransactionState** > **getTransactionState** () const

### 6.649.1 Constructor & Destructor Documentation

6.649.1.1 **activemq::state::ProducerState::ProducerState** ( const **Pointer**< **ProducerInfo** > & *info* )

6.649.1.2 virtual **activemq::state::ProducerState::~~ProducerState** ( ) [virtual]

### 6.649.2 Member Function Documentation

6.649.2.1 const **Pointer**<**ProducerInfo**>& **activemq::state::ProducerState::getInfo** ( ) const [inline]

6.649.2.2 **Pointer**<**TransactionState**> **activemq::state::ProducerState::getTransactionState** ( ) const

6.649.2.3 void **activemq::state::ProducerState::setTransactionState** ( const **Pointer**< **TransactionState** > & *transactionState* )

6.649.2.4 std::string **activemq::state::ProducerState::toString** ( ) const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ProducerState.h**

## 6.650 decaf::util::Properties Class Reference

Java-like properties class for mapping string names to string values.

```
#include <src/main/decaf/util/Properties.h>
```

## Public Member Functions

- **Properties** ()
- **Properties** (const **Properties** &src)
- virtual ~**Properties** ()
- **Properties** & **operator=** (const **Properties** &src)  
*Assignment Operator.*
- bool **isEmpty** () const  
*Returns true if the properties object is empty.*
- std::size\_t **size** () const
- const char \* **getProperty** (const std::string &name) const  
*Looks up the value for the given property.*
- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const  
*Looks up the value for the given property.*
- std::string **setProperty** (const std::string &name, const std::string &value)  
*Sets the value for a given property.*
- bool **hasProperty** (const std::string &name) const  
*Check to see if the Property exists in the set.*
- std::string **remove** (const std::string &name)  
*Removes the property with the given name.*
- std::vector< std::string > **propertyNames** () const  
*Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.*
- std::vector< std::pair< std::string, std::string > > **toArray** () const  
*Method that serializes the contents of the property map to an array.*
- void **copy** (const **Properties** &source)  
*Copies the contents of the given properties object to this one, if the given **Properties** (p. 3072) instance in NULL then this **List** (p. 2296) is not modified.*
- **Properties** \* **clone** () const  
*Clones this object.*
- void **clear** ()  
*Clears all properties from the map.*
- bool **equals** (const **Properties** &source) const  
*Test whether two **Properties** (p. 3072) objects are equivalent.*
- std::string **toString** () const  
*Formats the contents of the **Properties** (p. 3072) Object into a string that can be logged, etc.*
- void **load** (decaf::io::InputStream \*stream) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )  
*Reads a property list (key and element pairs) from the input byte stream.*
- void **load** (decaf::io::Reader \*reader) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )



*Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.*

- void **store** (decaf::io::OutputStream \*out, const std::string &comment) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

*Writes this property list (key and element pairs) in this **Properties** (p. 3072) table to the output stream in a format suitable for loading into a **Properties** (p. 3072) table using the load(InputStream) method.*

- void **store** (decaf::io::Writer \*writer, const std::string &comments) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

*Writes this property list (key and element pairs) in this **Properties** (p. 3072) table to the output character stream in a format that can be read by the load(Reader) method.*

## Protected Attributes

- decaf::lang::Pointer< **Properties** > defaults

*Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.*

### 6.650.1 Detailed Description

Java-like properties class for mapping string names to string values.

The **Properties** (p. 3072) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 3072) instance can contain an internal **Properties** (p. 3072) list that contains default values for keys not found in the **Properties** (p. 3072) **List** (p. 2296).

The **Properties** (p. 3072) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

## Since

1.0

### 6.650.2 Constructor & Destructor Documentation

6.650.2.1 decaf::util::Properties::Properties ( )

6.650.2.2 decaf::util::Properties::Properties ( const Properties & src )

6.650.2.3 virtual decaf::util::Properties::~~Properties ( ) [virtual]

### 6.650.3 Member Function Documentation

6.650.3.1 void decaf::util::Properties::clear ( )

Clears all properties from the map.

### 6.650.3.2 **Properties\*** `decaf::util::Properties::clone ( ) const`

Clones this object.

#### Returns

a replica of this object.

### 6.650.3.3 `void decaf::util::Properties::copy ( const Properties & source )`

Copies the contents of the given properties object to this one, if the given **Properties** (p. 3072) instance is NULL then this **List** (p. 2296) is not modified.

#### Parameters

<i>source</i>	The source properties object.
---------------	-------------------------------

### 6.650.3.4 `bool decaf::util::Properties::equals ( const Properties & source ) const`

Test whether two **Properties** (p. 3072) objects are equivalent.

Two **Properties** (p. 3072) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

#### Parameters

<i>source</i>	The <b>Properties</b> (p. 3072) object to compare this instance to.
---------------	---

#### Returns

true if the contents of the two **Properties** (p. 3072) objects are the same.

### 6.650.3.5 `const char* decaf::util::Properties::getProperty ( const std::string & name ) const`

Looks up the value for the given property.

#### Parameters

<i>name</i>	The name of the property to be looked up.
-------------	---

#### Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

6.650.3.6 `std::string decaf::util::Properties::getProperty ( const std::string & name, const std::string & defaultValue ) const`

Looks up the value for the given property.

#### Parameters

<i>name</i>	The name of the property to be looked up.
<i>defaultValue</i>	The value to be returned if the given property does not exist.

#### Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

6.650.3.7 `bool decaf::util::Properties::hasProperty ( const std::string & name ) const`

Check to see if the Property exists in the set.

#### Parameters

<i>name</i>	The property name to check for in this properties set.
-------------	--

#### Returns

true if property exists, false otherwise.

6.650.3.8 `bool decaf::util::Properties::isEmpty ( ) const`

Returns true if the properties object is empty.

#### Returns

true if empty

6.650.3.9 `void decaf::util::Properties::load ( decaf::io::InputStream  
* stream ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::NullPointerException )`

Reads a property list (key and element pairs) from the input byte stream.

The input stream is in a simple line-oriented format as specified in load(Reader) and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

#### Parameters

<i>stream</i>	The stream to read the properties data from.
---------------	--

**Exceptions**

<i>IOException</i>	if there is an error while reading from the stream.
<i>IllegalArgumentException</i>	if malformed data is found while reading the properties.
<i>NullPointerException</i>	if the passed stream is Null.

```
6.650.3.10 void decaf::util::Properties::load ( decaf::io::Reader
* reader ) throw ( decaf::io::IOException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::NullPointerException )
```

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

**Properties** (p. 3072) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character \. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab (" "), and form feed (" ") to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no affect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of 2n contiguous backslashes before a line terminator (or elsewhere) encodes n backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

```
\: \=
```

would be the two-character key "=". Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=' or ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

```
Truth = Beauty Truth:Beauty Truth :Beauty
```

As another example, the following three lines specify a single property:

```
fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango
```

The key is "fruits" and the associated element is: "apple, banana, pear, cantaloupe, watermelon, kiwi, mango"

Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

```
cheeses
```

specifies that the key is "cheeses" and the associated element is the empty string "".

Characters in keys and elements can be represented in escape sequences similar to those used for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.
- The method does not treat a backslash character, \, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence "\z" would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence "\b" as equivalent to the single character 'b'.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

#### Parameters

<i>reader</i>	The Reader that provides an character stream as input.
---------------	--

**Exceptions**

<i>IOException</i>	if there is an error while reading from the stream.
<i>IllegalArgumentException</i>	if malformed data is found while reading the properties.
<i>NullPointerException</i>	if the passed stream is Null.

**6.650.3.11 Properties& decaf::util::Properties::operator= ( const Properties & src )**

Assignment Operator.

**Parameters**

<i>src</i>	The <b>Properties</b> (p. 3072) list to copy to this <b>List</b> (p. 2296).
------------	---

**Returns**

a reference to this **List** (p. 2296) for use in chaining.

**6.650.3.12 std::vector<std::string> decaf::util::Properties::propertyNames ( ) const**

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

**Returns**

a set of keys in this property list where the key and its corresponding value are strings, including the keys in the default property list.

**6.650.3.13 std::string decaf::util::Properties::remove ( const std::string & name )**

Removes the property with the given name.

**Parameters**

<i>name</i>	The name of the property to remove.
-------------	-------------------------------------

**Returns**

the previous value of the property if set, or empty string.

**6.650.3.14 std::string decaf::util::Properties::setProperty ( const std::string & name, const std::string & value )**

Sets the value for a given property.

If the property already exists, overwrites the value.

#### Parameters

<i>name</i>	The name of the value to be written.
<i>value</i>	The value to be written.

#### Returns

the old value of the property or empty string if not set.

6.650.3.15 `std::size_t decaf::util::Properties::size ( ) const`

#### Returns

The number of **Properties** (p. 3072) in this **Properties** (p. 3072) Object.

6.650.3.16 `void decaf::util::Properties::store ( decaf::io::OutputStream *  
out, const std::string & comment ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::NullPointerException )`

Writes this property list (key and element pairs) in this **Properties** (p. 3072) table to the output stream in a format suitable for loading into a **Properties** (p. 3072) table using the load(InputStream) method.

**Properties** (p. 3072) from the defaults table of this **Properties** (p. 3072) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in store(Writer), with the following differences:

- The stream is written using the ISO 8859-1 character encoding.
- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value xxxx.
- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value xxxx.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

#### Parameters

<i>out</i>	The OutputStream instance to write the properties to.
<i>comment</i>	A description of these properties that is written to the output stream.

#### Exceptions

<i>IOException</i>	if there is an error while writing from the stream.
<i>NullPointerException</i>	if the passed stream is Null.

```
6.650.3.17 void decaf::util::Properties::store ( decaf::io::Writer * writer,
const std::string & comments ) throw ( decaf::io::IOException,
decaf::lang::exceptions::NullPointerException )
```

Writes this property list (key and element pairs) in this **Properties** (p. 3072) table to the output character stream in a format that can be read by the load(Reader) method.

**Properties** (p. 3072) from the defaults table of this **Properties** (p. 3072) table (if any) are not written out by this method.

If the comments argument is not empty, then an ASCII # character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed (

'), a carriage return ("), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character # or character ! then an ASCII # is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII # character, the current date and time (as if produced by the toString method of **Date** (p. 1633) for the current time), and a line separator as generated by the Writer.

Then every entry in this **Properties** (p. 3072) table is written out, one per line. For each entry the key string is written, then an ASCII =, then the associated element string. For the key, all space characters are written with a preceding \ character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding \ character. The key and element characters #, !, =, and : are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

#### Parameters

<i>writer</i>	The Writer instance to use to output the properties.
<i>comments</i>	A description of these properties that is written before writing the properties.

#### Exceptions

<i>IOException</i>	if there is an error while writing from the stream.
<i>NullPointerException</i>	if the passed stream is Null.

```
6.650.3.18 std::vector< std::pair< std::string, std::string > > decaf::util::Properties::toArray (
) const
```

Method that serializes the contents of the property map to an array.

#### Returns

list of pairs where the first is the name and the second is the value.



6.650.3.19 `std::string decaf::util::Properties::toString ( ) const`

Formats the contents of the **Properties** (p. 3072) Object into a string that can be logged, etc.

#### Returns

string value of this object.

### 6.650.4 Field Documentation

6.650.4.1 `decaf::lang::Pointer<Properties> decaf::util::Properties::defaults`  
[protected]

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Properties.h`

## 6.651 decaf::util::logging::PropertiesChangeListener Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 3072).

```
#include <src/main/decaf/util/logging/PropertiesChangeListener.h>
```

### Public Member Functions

- virtual `~PropertiesChangeListener ( )`
- virtual void `onPropertiesReset ( )=0`  
*Indicates that the **Properties** (p. 3072) have all been reset and should be considered to be back to their default values.*
- virtual void `onPropertyChanged (const std::string &name, const std::string &old-Value, const std::string &newValue)=0`  
*Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.*

### 6.651.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 3072).

#### Since

1.0

### 6.651.2 Constructor & Destructor Documentation

6.651.2.1 `virtual decaf::util::logging::PropertiesChangeListener::~~PropertiesChangeListener ( ) [inline, virtual]`

### 6.651.3 Member Function Documentation

6.651.3.1 `virtual void decaf::util::logging::PropertiesChangeListener::onPropertiesReset ( ) [pure virtual]`

Indicates that the **Properties** (p. 3072) have all been reset and should be considered to be back to their default values.

6.651.3.2 `virtual void decaf::util::logging::PropertiesChangeListener::onPropertyChanged ( const std::string & name, const std::string & oldValue, const std::string & newValue ) [pure virtual]`

Change Event, called when a property is changed, includes the name of the property that was changed along with it old and new values.

#### Parameters

<i>name</i>	The name of the Property that changed.
<i>oldValue</i>	The old Value of the Property.
<i>newValue</i>	The new Value of the Property.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/PropertiesChangeListener.h`

## 6.652 decaf::net::ProtocolException Class Reference

```
#include <src/main/decaf/net/ProtocolException.h>
```

Inheritance diagram for `decaf::net::ProtocolException`:

### Public Member Functions

- **ProtocolException** () throw ()  
*Default Constructor.*
- **ProtocolException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **ProtocolException** (const **ProtocolException** &ex) throw ()  
*Copy Constructor.*

- **ProtocolException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ProtocolException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ProtocolException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **ProtocolException** \* clone () const  
*Clones this exception.*
- virtual ~**ProtocolException** () throw ()

### 6.652.1 Constructor & Destructor Documentation

6.652.1.1 decaf::net::ProtocolException::ProtocolException ( ) throw () [inline]

Default Constructor.

6.652.1.2 decaf::net::ProtocolException::ProtocolException ( const Exception & ex ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

ex	An exception that should become this type of Exception
----	--

6.652.1.3 decaf::net::ProtocolException::ProtocolException ( const ProtocolException & ex ) throw () [inline]

Copy Constructor.

#### Parameters

ex	An exception that should become this type of Exception
----	--

6.652.1.4 decaf::net::ProtocolException::ProtocolException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.652.1.5 `decaf::net::ProtocolException::ProtocolException ( const std::exception * cause )  
throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.652.1.6 `decaf::net::ProtocolException::ProtocolException ( const char * file, const int  
lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.652.1.7 `virtual decaf::net::ProtocolException::~~ProtocolException ( ) throw ()  
[inline, virtual]`

**6.652.2 Member Function Documentation**

6.652.2.1 `virtual ProtocolException* decaf::net::ProtocolException::clone ( ) const  
[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 2105).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ProtocolException.h`

## 6.653 `decaf::security::PublicKey` Class Reference

A public key.

```
#include <src/main/decaf/security/PublicKey.h>
```

Inheritance diagram for `decaf::security::PublicKey`:

### Public Member Functions

- virtual `~PublicKey()`

#### 6.653.1 Detailed Description

A public key.

This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

#### 6.653.2 Constructor & Destructor Documentation

6.653.2.1 virtual `decaf::security::PublicKey::~~PublicKey()` [`inline`, `virtual`]

The documentation for this class was generated from the following file:

- `src/main/decaf/security/PublicKey.h`

## 6.654 `decaf::io::PushbackInputStream` Class Reference

A **PushbackInputStream** (p. 3086) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

```
#include <src/main/decaf/io/PushbackInputStream.h>
```

Inheritance diagram for `decaf::io::PushbackInputStream`:

## Public Member Functions

- **PushbackInputStream** (**InputStream** \*stream, bool **own**=false)

*Creates a **PushbackInputStream** (p. 3086) and saves its argument, the input stream in, for later use.*

- **PushbackInputStream** (**InputStream** \*stream, int bufSize, bool **own**=false) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Creates a **PushbackInputStream** (p. 3086) and saves its argument, the input stream in, for later use.*

- virtual ~**PushbackInputStream** ()

- void **unread** (unsigned char value) throw ( decaf::io::IOException )

*Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.*

- void **unread** (const unsigned char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.*

- void **unread** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.*

- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.*

*The default implementation of this method returns zero.*

**Returns**

*the number of bytes available on this input stream.*

**Exceptions**

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

- virtual long long **skip** (long long num) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )

*Skips over and discards n bytes of data from this input stream.*

*The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.*

*The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.*

**Parameters**

num	<i>The number of bytes to skip.</i>
-----	-------------------------------------

**Returns***total bytes skipped***Exceptions**

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
UnsupportedOperationException	<i>if the concrete stream class does not support skipping bytes.</i>

- virtual void **mark** (int readLimit)  
*Does nothing except throw an **IOException** (p. 2103).*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Does nothing except throw an **IOException** (p. 2103).*
- virtual bool **markSupported** () const  
*Does nothing except throw an **IOException** (p. 2103).*

**Protected Member Functions**

- virtual int **doReadByte** () throw ( decaf::io::IOException )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

**6.654.1 Detailed Description**

A **PushbackInputStream** (p. 3086) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.

This is useful in situations where it is convenient for a fragment of code to read an indefinite number of data bytes that are delimited by a particular byte value; after reading the terminating byte, the code fragment can "unread" it, so that the next read operation on the input stream will reread the byte that was pushed back. For example, bytes representing the characters constituting an identifier might be terminated by a byte representing an operator character; a method whose job is to read just an identifier can read until it sees the operator and then push the operator back to be re-read.

**Since**

1.0

**6.654.2 Constructor & Destructor Documentation**

**6.654.2.1** decaf::io::PushbackInputStream::PushbackInputStream ( **InputStream** \* *stream*, bool *own* = false )

Creates a **PushbackInputStream** (p. 3086) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

**Parameters**

<i>stream</i>	The <b>InputStream</b> (p. 2002) instance to wrap.
<i>Boolean</i>	value indicating if this <b>FilterInputStream</b> (p. 1854) owns the wrapped stream.

6.654.2.2 `decaf::io::PushbackInputStream::PushbackInputStream (   
 InputStream * stream, int bufSize, bool own = false ) throw (   
 decaf::lang::exceptions::IllegalArgumentException )`

Creates a **PushbackInputStream** (p. 3086) and saves its argument, the input stream in, for later use.

Initially, there is no pushed-back byte.

**Parameters**

<i>stream</i>	The <b>InputStream</b> (p. 2002) instance to wrap.
<i>bufSize</i>	The number of byte to allocate for pushback into this stream.
<i>Boolean</i>	value indicating if this <b>FilterInputStream</b> (p. 1854) owns the wrapped stream.

**Exceptions**

<i>IllegalArgumentException</i>	if the <i>bufSize</i> argument is < zero.
---------------------------------	---

6.654.2.3 `virtual decaf::io::PushbackInputStream::~~PushbackInputStream ( ) [virtual]`

**6.654.3 Member Function Documentation**

6.654.3.1 `virtual int decaf::io::PushbackInputStream::available ( ) const throw (   
 decaf::io::IOException ) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

**Returns**

the number of bytes available on this input stream.

**Exceptions**

<i>IOException</i> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------



Returns the sum of the number of pushed back bytes if any and the amount of bytes available in the underlying stream via a call to `available`.

Reimplemented from **decaf::io::FilterInputStream** (p. 1857).

```
6.654.3.2 virtual int decaf::io::PushbackInputStream::doReadArrayBounded ( unsigned char
    * buffer, int size, int offset, int length ) throw ( decaf::io::IOException,
    decaf::lang::exceptions::IndexOutOfBoundsException,
    decaf::lang::exceptions::NullPointerException ) [protected,
    virtual]
```

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

```
6.654.3.3 virtual int decaf::io::PushbackInputStream::doReadByte ( ) throw (
    decaf::io::IOException ) [protected, virtual]
```

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

```
6.654.3.4 virtual void decaf::io::PushbackInputStream::mark ( int readLimit ) [virtual]
```

Does nothing except throw an **IOException** (p. 2103).

Marks the current position in the stream. A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Calling `mark` on a closed stream instance should have no effect.

The default implementation of this method does nothing.

#### Parameters

<i>readLimit</i>	The max bytes read before marked position is invalid.
------------------	---

Reimplemented from **decaf::io::FilterInputStream** (p. 1858).

```
6.654.3.5 virtual bool decaf::io::PushbackInputStream::markSupported ( ) const
    [inline, virtual]
```

Does nothing except throw an **IOException** (p. 2103).

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

The default implementation of this method returns false.

**Returns**

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1859).

```
6.654.3.6  virtual void decaf::io::PushbackInputStream::reset (    ) throw (
            decaf::io::IOException ) [virtual]
```

Does nothing except throw an **IOException** (p. 2103).

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: \* If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 2103) might be thrown. \* If such an **IOException** (p. 2103) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset.

If the method markSupported returns false, then: \* The call to reset may throw an **IOException** (p. 2103). \* If an **IOException** (p. 2103) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

The default implementation of this method throws an **IOException** (p. 2103).

**Exceptions**

<b>IOException</b> (p. 2103)	if an I/O error occurs.
---------------------------------	-------------------------

Reimplemented from **decaf::io::FilterInputStream** (p. 1859).

```
6.654.3.7  virtual long long decaf::io::PushbackInputStream::skip
            ( long long num ) throw ( decaf::io::IOException,
            decaf::lang::exceptions::UnsupportedOperationException )
            [virtual]
```

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until num bytes have been read or the end of the stream has been reached.

Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

This method first skips bytes in the local pushed back buffer before attempting to complete the request by calling the underlying stream skip method with the remainder of bytes that needs to be skipped.

Reimplemented from **decaf::io::FilterInputStream** (p. 1860).

```
6.654.3.8 void decaf::io::PushbackInputStream::unread ( const unsigned
char * buffer, int size ) throw ( decaf::io::IOException,
decaf::lang::exceptions::IndexOutOfBoundsException,
decaf::lang::exceptions::NullPointerException )
```

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

#### Parameters

<i>buffer</i>	The bytes to copy to the front of push back buffer.
<i>size</i>	The size of the array to be copied.

#### Exceptions

<i>NullPointerException</i>	if the buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if the size value given is negative.
<b><i>IOException</i></b> (p. 2103)	if there is not enough space in the pushback buffer or this stream has already been closed.

6.654.3.9 `void decaf::io::PushbackInputStream::unread ( const unsigned char *  
buffer, int size, int offset, int length ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException )`

Pushes back the given array of bytes, the bytes are copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back bytes.

#### Parameters

<i>buffer</i>	The bytes to copy to the front of push back buffer.
<i>size</i>	The size of the array to be copied.
<i>offset</i>	The position in the buffer to start copying from.
<i>length</i>	The number of bytes to push back from the passed buffer.

#### Exceptions

<i>NullPointerException</i>	if the buffer passed is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length is greater than the buffer size.
<b><i>IOException</i></b> (p. 2103)	if there is not enough space in the pushback buffer or this stream has already been closed.

6.654.3.10 `void decaf::io::PushbackInputStream::unread ( unsigned char value ) throw ( decaf::io::IOException )`

Pushes back the given byte, the byte is copied to the front of the pushback buffer, future calls to read start reading from the beginning of these pushed back byte.

#### Parameters

<i>value</i>	The byte that is to be placed at the front of the push back buffer.
--------------	---

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if there is not enough space in the pushback buffer or this stream has already been closed.
--	---

The documentation for this class was generated from the following file:

- src/main/decaf/io/**PushbackInputStream.h**

## 6.655 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

```
#include <src/main/cms/Queue.h>
```

Inheritance diagram for cms::Queue:

## Public Member Functions

- virtual `~Queue()`
- virtual `std::string getQueueName() const` `=0` throw ( `CMSEException` )

*Gets the name of this queue.*

### 6.655.1 Detailed Description

An interface encapsulating a provider-specific queue name.

Messages sent to a **Queue** (p. 3093) are sent to a Single Subscriber on that **Queue** (p. 3093) **Destination** (p. 1688). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 2493) in a **Queue** (p. 3093) is not defined by the CMS API, consult your Provider documentation for this information.

#### Since

1.0

### 6.655.2 Constructor & Destructor Documentation

6.655.2.1 virtual `cms::Queue::~Queue()` `[inline, virtual]`

### 6.655.3 Member Function Documentation

6.655.3.1 virtual `std::string cms::Queue::getQueueName() const` throw ( `CMSEException` )  
`[pure virtual]`

Gets the name of this queue.

#### Returns

The queue name.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- If an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQQueue** (p. 456).

The documentation for this class was generated from the following file:

- `src/main/cms/Queue.h`

## 6.656 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

```
#include <src/main/decaf/util/Queue.h>
```

Inheritance diagram for decaf::util::Queue< E >:

### Public Member Functions

- virtual **~Queue** ()
- virtual bool **offer** (const E &value)=0 throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into the queue provided that the condition allows such an operation.*
- virtual bool **poll** (E &result)=0  
*Gets and removes the element in the head of the queue.*
- virtual E **remove** ()=0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets and removes the element in the head of the queue.*
- virtual bool **peek** (E &result) const =0  
*Gets but not removes the element in the head of the queue.*
- virtual E **element** () const =0 throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets but not removes the element in the head of the queue.*

### 6.656.1 Detailed Description

```
template<typename E>class decaf::util::Queue< E >
```

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

**Queue** (p. 3094) does not provide blocking queue methods, which will block until the operation of the method is allowed. BlockingQueue interface defines such methods.

Unlike the Java **Queue** (p. 3094) interface the methods of this class cannot return null to indicate that a **Queue** (p. 3094) is empty since null has no meaning for elements such as classes, structs and primitive types and cannot be used in a meaningful way to check for an empty queue. Methods that would have returned null in the Java **Queue** (p. 3094) interface have been altered to return a boolean value indicating if the operation succeeded and take single argument that is a reference to the location where the returned

value is to be assigned. This implies that elements in the **Queue** (p. 3094) must be *assignable* in order to utilize these methods.

### Since

1.0

## 6.656.2 Constructor & Destructor Documentation

6.656.2.1 `template<typename E> virtual decaf::util::Queue< E >::~~Queue ( )`  
`[inline, virtual]`

## 6.656.3 Member Function Documentation

6.656.3.1 `template<typename E> virtual E decaf::util::Queue< E >::element ( ) const`  
`throw ( decaf::lang::exceptions::NoSuchElementException ) [pure`  
`virtual]`

Gets but not removes the element in the head of the queue.

Throws a NoSuchElementException if there is no element in the queue.

### Returns

the element in the head of the queue.

### Exceptions

<i>NoSuchElementException</i>	if there is no element in the queue.
-------------------------------	--------------------------------------

Implemented in **decaf::util::AbstractQueue< E >** (p. 166).

6.656.3.2 `template<typename E> virtual bool decaf::util::Queue< E >::offer ( const`  
`E & value ) throw ( decaf::lang::exceptions::NullPointerException,`  
`decaf::lang::exceptions::IllegalArgumentException ) [pure`  
`virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

### Parameters

<i>value</i>	the specified element to insert into the queue.
--------------	---

### Returns

true if the operation succeeds and false if it fails.

**Exceptions**

<i>NullPointerException</i>	if the <b>Queue</b> (p. 3094) implementation does not allow Null values to be inserted into the <b>Queue</b> (p. 3094).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3666), and **decaf::util::PriorityQueue< E >** (p. 2981).

Referenced by **decaf::util::AbstractQueue< E >::add()**.

6.656.3.3 **template<typename E> virtual bool decaf::util::Queue< E >::peek ( E & result )**  
**const** [pure virtual]

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

**Parameters**

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

**Returns**

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::PriorityQueue< E >** (p. 2982).

Referenced by **decaf::util::AbstractQueue< E >::element()**.

6.656.3.4 **template<typename E> virtual bool decaf::util::Queue< E >::poll ( E & result )**  
[pure virtual]

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 3094) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

**Parameters**

<i>result</i>	Reference to an instance of the contained type to assigned the removed value to.
---------------	--

**Returns**

true if the element at the head of the queue was removed and assigned to the result parameter.



Implemented in **decaf::util::concurrent::SynchronousQueue< E >** (p. 3667), and **decaf::util::PriorityQueue< E >** (p. 2982).

Referenced by **decaf::util::AbstractQueue< E >::clear()**, and **decaf::util::AbstractQueue< E >::remove()**.

```
6.656.3.5  template<typename E> virtual E decaf::util::Queue< E >::remove ( )
            throw ( decaf::lang::exceptions::NoSuchElementException ) [pure
            virtual]
```

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** if there is no element in the queue.

### Returns

the element in the head of the queue.

### Exceptions

<i>NoSuchElementException</i>	if there is no element in the queue.
-------------------------------	--------------------------------------

Implemented in **decaf::util::AbstractQueue< E >** (p. 167), and **decaf::util::PriorityQueue< E >** (p. 2982).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Queue.h**

## 6.657 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p. 3093) without removing them.

```
#include <src/main/cms/QueueBrowser.h>
```

Inheritance diagram for cms::QueueBrowser:

### Public Member Functions

- virtual **~QueueBrowser** ()
- virtual const **Queue** \* **getQueue** () const =0 throw ( cms::CMSEException )
- virtual std::string **getMessageSelector** () const =0 throw ( cms::CMSEException )
- virtual **cms::MessageEnumeration** \* **getEnumeration** ()=0 throw ( cms::CMSEException )

*Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 3093) in the order that a client would receive them.*

### 6.657.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p. 3093) without removing them.

To browse the contents of the **Queue** (p. 3093) the client calls the `getEnumeration` method to retrieve a new instance of a **Queue** (p. 3093) Enumerator. The client then calls the `hasMoreMessages` method of the Enumeration, if it returns true the client can safely call the `nextMessage` method of the Enumeration instance.

```
Enumeration* enumeration = queueBrowser->getEnumeration() (p. 3099);
while( enumeration->hasMoreMessages() ) { cms::Message (p. 2493)* message =
enumeration->nextMessage();
// ... Do something with the Message (p. 2493).
delete message; }
```

#### Since

1.1

### 6.657.2 Constructor & Destructor Documentation

6.657.2.1 `virtual cms::QueueBrowser::~QueueBrowser ( ) [inline, virtual]`

### 6.657.3 Member Function Documentation

6.657.3.1 `virtual cms::MessageEnumeration* cms::QueueBrowser::getEnumeration ( ) throw ( cms::CMSEException ) [pure virtual]`

Gets a pointer to an Enumeration object for browsing the Messages currently in the **Queue** (p. 3093) in the order that a client would receive them.

The pointer returned is owned by the browser and should not be deleted by the client application.

#### Returns

a pointer to a **Queue** (p. 3093) Enumeration, this Pointer is owned by the **Queue-Browser** (p. 3098) and should not be deleted by the client.

#### Exceptions

<b>CMSEException</b> (p. 1130)	if an internal error occurs.
-----------------------------------	------------------------------

Implemented in `activemq::core::ActiveMQQueueBrowser` (p. 458).

6.657.3.2 virtual std::string cms::QueueBrowser::getMessageSelector ( ) const throw ( cms::CMSEException ) [pure virtual]

Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions

<b>CMSEException</b> (p. 1130)	if an internal error occurs.
-----------------------------------	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 459).

6.657.3.3 virtual const Queue\* cms::QueueBrowser::getQueue ( ) const throw ( cms::CMSEException ) [pure virtual]

Returns

the **Queue** (p. 3093) that this browser is listening on.

Exceptions

<b>CMSEException</b> (p. 1130)	if an internal error occurs.
-----------------------------------	------------------------------

Implemented in **activemq::core::ActiveMQQueueBrowser** (p. 459).

The documentation for this class was generated from the following file:

- src/main/cms/**QueueBrowser.h**

6.658 decaf::util::Random Class Reference

**Random** (p. 3100) Value Generator which is used to generate a stream of pseudorandom numbers.

```
#include <src/main/decaf/util/Random.h>
```

Inheritance diagram for decaf::util::Random:

Public Member Functions

- **Random ()**  
*Construct a random generator with the current time of day in milliseconds as the initial state.*

- **Random** (unsigned long long seed)  
*Construct a random generator with the given `seed` as the initial state.*
- bool **nextBoolean** ()  
*Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.*
- double **nextDouble** ()  
*Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.*
- float **nextFloat** ()  
*Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.*
- double **nextGaussian** ()  
*Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the polar method of G.*
- int **nextInt** ()  
*Generates a uniformly distributed 32-bit `int` value from the this random number sequence.*
- int **nextInt** (int n)  
*Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).*
- long long **nextLong** ()  
*Generates a uniformly distributed 64-bit `int` value from the this random number sequence.*
- virtual void **nextBytes** (std::vector< unsigned char > &buf)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*
- virtual void **nextBytes** (unsigned char \*buf, int size)  
*Modifies the byte array by a random sequence of bytes generated by this random number generator.*
- virtual void **setSeed** (unsigned long long seed)  
*Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.*

## Protected Member Functions

- virtual int **next** (int bits)  
*Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.*

### 6.658.1 Detailed Description

**Random** (p.3100) Value Generator which is used to generate a stream of pseudorandom numbers.

The algorithms implemented by class **Random** (p.3100) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

**Since**

1.0

**6.658.2 Constructor & Destructor Documentation****6.658.2.1 decaf::util::Random::Random ( )**

Construct a random generator with the current time of day in milliseconds as the initial state.

**See also****setSeed** (p. 3105)**6.658.2.2 decaf::util::Random::Random ( unsigned long long seed )**

Construct a random generator with the given `seed` as the initial state.

**Parameters**

<code>seed</code>	the seed that will determine the initial state of this random number generator
-------------------	--

**See also****setSeed** (p. 3105)**6.658.3 Member Function Documentation****6.658.3.1 virtual int decaf::util::Random::next ( int bits ) [protected, virtual]**

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

**Returns**`int` a pseudo-random generated `int` number**Parameters**

<code>bits</code>	number of bits of the returned value
-------------------	--------------------------------------

**See also****nextBytes** (p. 3103)**nextDouble** (p. 3104)**nextFloat** (p. 3104)**nextInt()** (p. 3104)

**nextInt(int)** (p. 3105)  
**nextGaussian** (p. 3104)  
**nextLong** (p. 3105)

Reimplemented in **decaf::security::SecureRandom** (p. 3272).

#### 6.658.3.2 `bool decaf::util::Random::nextBoolean ( )`

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

#### Returns

boolean a pseudo-random, uniformly distributed boolean value

#### 6.658.3.3 `virtual void decaf::util::Random::nextBytes ( unsigned char * buf, int size )` `[virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

#### Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

#### See also

**next** (p. 3102)

#### Exceptions

<i>NullPointerException</i>	if buff is NULL
<i>IllegalArgumentException</i>	if size is negative

Reimplemented in **decaf::security::SecureRandom** (p. 3273).

#### 6.658.3.4 `virtual void decaf::util::Random::nextBytes ( std::vector< unsigned char > & buf )` `[virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

#### Parameters

<i>buf</i>	non-null array to contain the new random bytes
------------	--

**See also**

**next** (p. 3102)

Reimplemented in **decaf::security::SecureRandom** (p. 3273).

**6.658.3.5 double decaf::util::Random::nextDouble ( )**

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

**Returns**

double

**See also**

**nextFloat** (p. 3104)

**6.658.3.6 float decaf::util::Random::nextFloat ( )**

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

**Returns**

float a random float number between 0.0 and 1.0

**See also**

**nextDouble** (p. 3104)

**6.658.3.7 double decaf::util::Random::nextGaussian ( )**

Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of 1.0 using the *polar method* of G.

E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

**Returns**

double

**See also**

**nextDouble** (p. 3104)

### 6.658.3.8 `int decaf::util::Random::nextInt ( )`

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

#### Returns

`int` uniformly distributed `int` value

#### See also

`next` (p. 3102)

`nextLong` (p. 3105)

### 6.658.3.9 `int decaf::util::Random::nextInt ( int n )`

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

#### Parameters

<code>n</code>	The <code>int</code> value that defines the max value of the return.
----------------	--

#### Returns

the next pseudo random `int` value.

#### Exceptions

<i><code>IllegalArgumentException</code></i>	if <code>n</code> is less than or equal to zero.
--	--

### 6.658.3.10 `long long decaf::util::Random::nextLong ( )`

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

#### Returns

64-bit `int` random number

#### See also

`next` (p. 3102)

`nextInt()` (p. 3104)

`nextInt(int)` (p. 3105)



6.658.3.11 `virtual void decaf::util::Random::setSeed ( unsigned long long seed )`  
`[virtual]`

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

#### Parameters

<code>seed</code>	the seed that alters the state of the random number generator
-------------------	---

#### See also

`next` (p. 3102)  
`Random()` (p. 3102)  
`#Random(long)`

Reimplemented in `decaf::security::SecureRandom` (p. 3274).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Random.h`

## 6.659 decaf::lang::Readable Class Reference

A **Readable** (p. 3106) is a source of characters.

```
#include <src/main/decaf/lang/Readable.h>
```

Inheritance diagram for `decaf::lang::Readable`:

### Public Member Functions

- `virtual ~Readable ()`
- `virtual int read (decaf::nio::CharBuffer *charBuffer)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )`

*Attempts to read characters into the specified character buffer.*

### 6.659.1 Detailed Description

A **Readable** (p. 3106) is a source of characters.

Characters from a **Readable** (p. 3106) are made available to callers of the `read` method via a `CharBuffer`.

#### Since

1.0

## 6.659.2 Constructor & Destructor Documentation

6.659.2.1 `virtual decaf::lang::Readable::~~Readable ( ) [inline, virtual]`

## 6.659.3 Member Function Documentation

6.659.3.1 `virtual int decaf::lang::Readable::read ( decaf::nio::CharBuffer * charBuffer ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException ) [pure virtual]`

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

### Parameters

<i>charBuffer</i>	The Buffer to read Characters into.
-------------------	-------------------------------------

### Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

### Exceptions

<i>IOException</i>	- if an I/O error occurs
<i>NullPointerException</i>	- if buffer is NULL.
<i>ReadOnlyBufferException</i>	- if <i>charBuffer</i> is a read only buffer

Implemented in **decaf::io::Reader** (p.3113).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Readable.h`

## 6.660 activemq::transport::inactivity::ReadChecker Class Reference

Runnable class that is used by the {.

```
#include <src/main/activemq/transport/inactivity/ReadChecker.h>
```

Inheritance diagram for `activemq::transport::inactivity::ReadChecker`:

### Public Member Functions

- **ReadChecker** (**InactivityMonitor** \*parent)

- virtual `~ReadChecker ()`
- virtual void `run ()`

*Run method - called by the Thread class in the context of the thread.*

### 6.660.1 Detailed Description

Runnable class that is used by the {.

#### See also

**InactivityMonitor** (p. 1964)} class the check for timeouts related to **transport** (p. 99) reads.

#### Since

3.1

### 6.660.2 Constructor & Destructor Documentation

6.660.2.1 `activemq::transport::inactivity::ReadChecker::ReadChecker ( InactivityMonitor * parent )`

6.660.2.2 `virtual activemq::transport::inactivity::ReadChecker::~~ReadChecker ( )`  
[virtual]

### 6.660.3 Member Function Documentation

6.660.3.1 `virtual void activemq::transport::inactivity::ReadChecker::run ( )` [virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3265).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/ReadChecker.h`

## 6.661 decaf::io::Reader Class Reference

```
#include <src/main/decaf/io/Reader.h>
```

Inheritance diagram for decaf::io::Reader:

## Public Member Functions

- virtual **~Reader** ()
- virtual void **mark** (int readAheadLimit) throw ( decaf::io::IOException )  
*Marks the present position in the stream.*
- virtual bool **markSupported** () const  
*Tells whether this stream supports the **mark()** (p. 3111) operation.*
- virtual bool **ready** () const throw ( decaf::io::IOException )  
*Tells whether this stream is ready to be read.*
- virtual void **reset** () throw ( decaf::io::IOException )  
*Resets the stream.*
- virtual long long **skip** (long long count) throw ( decaf::io::IOException )  
*Skips characters.*
- virtual int **read** (std::vector< char > &buffer) throw ( decaf::io::IOException )  
*Reads characters into an array.*
- virtual int **read** (char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Reads characters into an array, the method will attempt to read as much data as the size of the array.*
- virtual int **read** (char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Reads characters into a portion of an array.*
- virtual int **read** () throw ( decaf::io::IOException )  
*Reads a single character.*
- virtual int **read** (decaf::nio::CharBuffer \*charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )  
*Attempts to read characters into the specified character buffer.*

## Protected Member Functions

- **Reader** ()
- virtual int **doReadArrayBounded** (char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Override this method to customize the functionality of the method **read( unsigned char\* buffer, int size, int offset, int length )**.*
- virtual int **doReadVector** (std::vector< char > &buffer) throw ( decaf::io::IOException )  
*Override this method to customize the functionality of the method **read( std::vector<char> &buffer )** (p. 3113).*
- virtual int **doReadArray** (char \*buffer, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )

*Override this method to customize the functionality of the method `read( char* buffer, std::size_t length )`.*

- virtual int **doReadChar** ( ) throw ( decaf::io::IOException )

*Override this method to customize the functionality of the method `read()` (p. 3112).*

- virtual int **doReadCharBuffer** (decaf::nio::CharBuffer \*charBuffer) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )

*Override this method to customize the functionality of the method `read( CharBuffer* charBuffer )`.*

### 6.661.1 Constructor & Destructor Documentation

6.661.1.1 decaf::io::Reader::Reader ( ) [protected]

6.661.1.2 virtual decaf::io::Reader::~~Reader ( ) [virtual]

### 6.661.2 Member Function Documentation

6.661.2.1 virtual int decaf::io::Reader::doReadArray ( char \* buffer, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
[protected, virtual]

Override this method to customize the functionality of the method `read( char* buffer, std::size_t length )`.

6.661.2.2 virtual int decaf::io::Reader::doReadArrayBounded ( char \* buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
[protected, pure virtual]

Override this method to customize the functionality of the method `read( unsigned char* buffer, int size, int offset, int length )`.

All subclasses must override this method to provide the basic **Reader** (p.3108) functionality.

Implemented in **decaf::io::InputStreamReader** (p.2015).

6.661.2.3 virtual int decaf::io::Reader::doReadChar ( ) throw ( decaf::io::IOException )  
[protected, virtual]

Override this method to customize the functionality of the method `read()` (p. 3112).

```
6.661.2.4 virtual int decaf::io::Reader::doReadCharBuffer ( decaf::nio::CharBuffer
* charBuffer ) throw ( decaf::io::IOException,
decaf::lang::exceptions::NullPointerException,
decaf::nio::ReadOnlyBufferException ) [protected, virtual]
```

Override this method to customize the functionality of the method `read( CharBuffer* charBuffer )`.

```
6.661.2.5 virtual int decaf::io::Reader::doReadVector ( std::vector< char > & buffer ) throw (
decaf::io::IOException ) [protected, virtual]
```

Override this method to customize the functionality of the method `read( std::vector<char>& buffer )` (p. 3113).

```
6.661.2.6 virtual void decaf::io::Reader::mark ( int readAheadLimit ) throw (
decaf::io::IOException ) [virtual]
```

Marks the present position in the stream.

Subsequent calls to `reset()` (p. 3114) will attempt to reposition the stream to this point. Not all character-input streams support the `mark()` (p. 3111) operation.

#### Parameters

<i>readAheadLimit</i>	Limit on the number of characters that may be read while still preserving the mark. After reading this many characters, attempting to reset the stream may fail.
-----------------------	--

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs, or the stream does not support mark.
---------------------------------	--

```
6.661.2.7 virtual bool decaf::io::Reader::markSupported ( ) const [inline,
virtual]
```

Tells whether this stream supports the `mark()` (p. 3111) operation.

The default implementation always returns false. Subclasses should override this method.

#### Returns

true if and only if this stream supports the mark operation.

6.661.2.8 `virtual int decaf::io::Reader::read ( char * buffer, int size,  
int offset, int length ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IndexOutOfBoundsException,  
decaf::lang::exceptions::NullPointerException ) [virtual]`

Reads characters into a portion of an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

#### Parameters

<i>buffer</i>	The target char buffer.
<i>size</i>	The size in bytes of the target buffer.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The maximum number of bytes to read.

#### Returns

The number of bytes read or -1 if the end of stream is reached.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	thrown if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if the offset + length is greater than the array size.

6.661.2.9 `virtual int decaf::io::Reader::read ( ) throw ( decaf::io::IOException )  
[virtual]`

Reads a single character.

This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.

Subclasses that intend to support efficient single-character input should override this method.

#### Returns

The character read, as an integer in the range 0 to 65535 (0x00-0xffff), or -1 if the end of the stream has been reached.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	thrown if an I/O error occurs.
--	--------------------------------

6.661.2.10 `virtual int decaf::io::Reader::read ( char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )` [virtual]

Reads characters into an array, the method will attempt to read as much data as the size of the array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

#### Parameters

<i>buffer</i>	The target char buffer.
<i>size</i>	The size in bytes of the target buffer.

#### Returns

The number of bytes read or -1 if the end of stream is reached.

#### Exceptions

<b>IOException</b> (p. 2103)	thrown if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.

6.661.2.11 `virtual int decaf::io::Reader::read ( decaf::nio::CharBuffer * charBuffer ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::nio::ReadOnlyBufferException )` [virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

#### Parameters

<i>charBuffer</i>	The Buffer to read Characters into.
-------------------	-------------------------------------

#### Returns

The number of char values added to the buffer, or -1 if this source of characters is at its end

#### Exceptions

<b>IOException</b> (p. 2103)	- if an I/O error occurs
<i>NullPointerException</i>	- if buffer is NULL.
<i>ReadOnlyBufferException</i>	- if charBuffer is a read only buffer

Implements **decaf::lang::Readable** (p. 3107).



6.661.2.12 `virtual int decaf::io::Reader::read ( std::vector< char > & buffer ) throw ( decaf::io::IOException ) [virtual]`

Reads characters into an array.

This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.

#### Parameters

<i>buffer</i>	The buffer to read characters into.
---------------	-------------------------------------

#### Returns

The number of characters read, or -1 if the end of the stream has been reached

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	thrown if an I/O error occurs.
--	--------------------------------

6.661.2.13 `virtual bool decaf::io::Reader::ready ( ) const throw ( decaf::io::IOException ) [virtual]`

Tells whether this stream is ready to be read.

#### Returns

True if the next **read()** (p. 3112) is guaranteed not to block for input, false otherwise. Note that returning false does not guarantee that the next read will block.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

Reimplemented in **decaf::io::InputStreamReader** (p. 2015).

6.661.2.14 `virtual void decaf::io::Reader::reset ( ) throw ( decaf::io::IOException ) [virtual]`

Resets the stream.

If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the **reset()** (p.3114) operation, and some support **reset()** (p.3114) without supporting **mark()** (p.3111).

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

6.661.2.15 `virtual long long decaf::io::Reader::skip ( long long count ) throw ( decaf::io::IOException )` [virtual]

Skips characters.

This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.

**Parameters**

<i>count</i>	The number of character to skip.
--------------	----------------------------------

**Returns**

the number of Character actually skipped.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Reader.h**

**6.662 decaf::nio::ReadOnlyBufferException Class Reference**

```
#include <src/main/decaf/nio/ReadOnlyBufferException.h>
```

Inheritance diagram for decaf::nio::ReadOnlyBufferException:

**Public Member Functions**

- **ReadOnlyBufferException** () throw ()  
*Default Constructor.*
- **ReadOnlyBufferException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **ReadOnlyBufferException** (const **ReadOnlyBufferException** &ex) throw ()  
*Copy Constructor.*
- **ReadOnlyBufferException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **ReadOnlyBufferException** (const std::exception \***cause**) throw ()

*Constructor.*

- **ReadOnlyBufferException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor.*

- virtual **ReadOnlyBufferException** \* **clone** () const

*Clones this exception.*

- virtual ~**ReadOnlyBufferException** () throw ()

## 6.662.1 Constructor & Destructor Documentation

6.662.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( ) throw ()  
[inline]

Default Constructor.

6.662.1.2 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.662.1.3 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const ReadOnlyBufferException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.662.1.4 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw ()  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.662.1.5 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.662.1.6 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.662.1.7 `virtual decaf::nio::ReadOnlyBufferException::~~ReadOnlyBufferException ( ) throw () [inline, virtual]`

**6.662.2 Member Function Documentation**

6.662.2.1 `virtual ReadOnlyBufferException* decaf::nio::ReadOnlyBufferException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::exceptions::UnsupportedOperationException** (p.3852).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/ReadOnlyBufferException.h`

## 6.663 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 3117) maintains a pair of associated locks, one for read-only operations and one for writing.

```
#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>
```

### Public Member Functions

- virtual **~ReadWriteLock** ()
- virtual **Lock & readLock** ()=0  
*Returns the lock used for reading.*
- virtual **Lock & writeLock** ()=0  
*Returns the lock used for writing.*

#### 6.663.1 Detailed Description

A **ReadWriteLock** (p. 3117) maintains a pair of associated locks, one for read-only operations and one for writing.

The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 3117) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 2336) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of code. Ultimately, only profiling and

measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

- \* Determining whether to grant the read lock or the write lock, when both readers and writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-lived as expected. Fair, or "in-order" implementations are also possible.
- \* Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency.
- \* Determining whether the locks are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant?
- \* Can the write lock be downgraded to a read lock without allowing an intervening writer?
- Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

## Since

1.0

## 6.663.2 Constructor & Destructor Documentation

6.663.2.1 `virtual decaf::util::concurrent::locks::ReadWriteLock::~~ReadWriteLock ( )`  
[inline, virtual]

## 6.663.3 Member Function Documentation

6.663.3.1 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock ( )` [pure virtual]

Returns the lock used for reading.

## Returns

the lock used for reading.

6.663.3.2 `virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock ( )`  
[pure virtual]

Returns the lock used for writing.

## 6.664 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference 3129

### Returns

the lock used for writing.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**ReadWriteLock.h**

## 6.664 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

### Public Member Functions

- **ReceiveExecutor** (**CmsTemplate** \*parent, **cms::Destination** \*destination, const std::string &selector, bool noLocal)
- virtual ~**ReceiveExecutor** ()
- virtual void **doInCms** (**cms::Session** \*session) throw (cms::CMSException)  
*Execute any number of operations against the supplied CMS session.*
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \*session AMQCPP\_UNUSED) throw ( cms::CMSException )
- **cms::Message** \* **getMessage** ()

### Protected Member Functions

- **ReceiveExecutor** (const **ReceiveExecutor** &)
- **ReceiveExecutor** & **operator=** (const **ReceiveExecutor** &)

### Protected Attributes

- **cms::Destination** \* destination
- std::string selector
- bool noLocal
- **cms::Message** \* message
- **CmsTemplate** \* parent

### 6.664.1 Constructor & Destructor Documentation

- 6.664.1.1 `activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor ( const ReceiveExecutor & ) [inline, protected]`
- 6.664.1.2 `activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor ( CmsTemplate * parent, cms::Destination * destination, const std::string & selector, bool noLocal ) [inline]`
- 6.664.1.3 `virtual activemq::cmsutil::CmsTemplate::ReceiveExecutor::~ReceiveExecutor ( ) [inline, virtual]`

### 6.664.2 Member Function Documentation

- 6.664.2.1 `virtual void activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms ( cms::Session * session ) throw (cms::CMSEException) [virtual]`

Execute any number of operations against the supplied CMS session.

#### Parameters

<i>session</i>	the CMS Session
----------------	-----------------

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1130)	if thrown by CMS API methods
---	------------------------------

Implements `activemq::cmsutil::SessionCallback` (p. 3320).

- 6.664.2.2 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination ( cms::Session *session AMQCPP_UNUSED ) throw ( cms::CMSEException ) [inline, virtual]`
- 6.664.2.3 `cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage ( ) [inline]`
- 6.664.2.4 `ReceiveExecutor& activemq::cmsutil::CmsTemplate::ReceiveExecutor::operator= ( const ReceiveExecutor & ) [inline, protected]`

### 6.664.3 Field Documentation

- 6.664.3.1 `cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination [protected]`
- 6.664.3.2 `cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::message [protected]`



6.664.3.3 `bool activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal`  
[protected]

6.664.3.4 `CmsTemplate* activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent`  
[protected]

6.664.3.5 `std::string activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.665 activemq::core::RedeliveryPolicy Class Reference

Interface for a **RedeliveryPolicy** (p. 3121) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

```
#include <src/main/activemq/core/RedeliveryPolicy.h>
```

Inheritance diagram for `activemq::core::RedeliveryPolicy`:

### Public Member Functions

- virtual `~RedeliveryPolicy ()`
- virtual double `getBackOffMultiplier ()` const =0
- virtual void `setBackOffMultiplier (double value)=0`  
*Sets the Back-Off Multiplier for Message Redelivery.*
- virtual short `getCollisionAvoidancePercent ()` const =0
- virtual void `setCollisionAvoidancePercent (short value)=0`
- virtual long long `getInitialRedeliveryDelay ()` const =0  
*Gets the initial time that redelivery of messages is delayed.*
- virtual void `setInitialRedeliveryDelay (long long value)=0`  
*Sets the initial time that redelivery will be delayed.*
- virtual int `getMaximumRedeliveries ()` const =0  
*Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.*
- virtual void `setMaximumRedeliveries (int maximumRedeliveries)=0`  
*Sets the Maximum allowable redeliveries for a Message.*
- virtual long long `getRedeliveryDelay (long long previousDelay)=0`  
*Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.*
- virtual bool `isUseCollisionAvoidance ()` const =0
- virtual void `setUseCollisionAvoidance (bool value)=0`

- virtual bool **isUseExponentialBackOff** () const =0
- virtual void **setUseExponentialBackOff** (bool value)=0
- virtual **RedeliveryPolicy** \* **clone** () const =0  
*Create a copy of this Policy and return it.*
- virtual void **configure** (const **decaf::util::Properties** &properties)  
*Checks the supplied properties object for properties matching the configurable settings of this class.*

### Static Public Attributes

- static const long long **NO\_MAXIMUM\_REDELIVERIES**

### Protected Member Functions

- **RedeliveryPolicy** ()

#### 6.665.1 Detailed Description

Interface for a **RedeliveryPolicy** (p. 3121) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.

#### Since

3.2.0

#### 6.665.2 Constructor & Destructor Documentation

6.665.2.1 **activemq::core::RedeliveryPolicy::RedeliveryPolicy** ( ) [protected]

6.665.2.2 **virtual activemq::core::RedeliveryPolicy::~~RedeliveryPolicy** ( ) [virtual]

#### 6.665.3 Member Function Documentation

6.665.3.1 **virtual RedeliveryPolicy\*** **activemq::core::RedeliveryPolicy::clone** ( ) const  
 [pure virtual]

Create a copy of this Policy and return it.

#### Returns

pointer to a new **RedeliveryPolicy** (p. 3121) that is a copy of this one.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1645).

6.665.3.2 `virtual void activemq::core::RedeliveryPolicy::configure ( const decaf::util::Properties & properties ) [virtual]`

Checks the supplied properties object for properties matching the configurable settings of this class.

The default implementation looks for properties named with the prefix `cms.RedeliveryPolicy.XXX` where XXX is the name of a property with a public setter method. For instance `cms.RedeliveryPolicy.useExponentialBackOff` will be used to set the value of the use exponential back off toggle.

Subclasses can override this method to add more configuration options or to exclude certain parameters from being set via the properties object.

#### Parameters

<i>properties</i>	The Properties object used to configure this object.
-------------------	--

#### Exceptions

<i>NumberFormatException</i>	if a property that is numeric cannot be converted
<i>IllegalArgumentException</i>	if a property can't be converted to the correct type.

6.665.3.3 `virtual double activemq::core::RedeliveryPolicy::getBackOffMultiplier ( ) const [pure virtual]`

#### Returns

The value of the Back-Off Multiplier for Message Redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1645).

6.665.3.4 `virtual short activemq::core::RedeliveryPolicy::getCollisionAvoidancePercent ( ) const [pure virtual]`

#### Returns

the currently set Collision Avoidance percentage.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1645).

6.665.3.5 `virtual long long activemq::core::RedeliveryPolicy::getInitialRedeliveryDelay ( ) const [pure virtual]`

Gets the initial time that redelivery of messages is delayed.

#### Returns

the time in milliseconds that redelivery is delayed initially.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1646).

6.665.3.6 `virtual int activemq::core::RedeliveryPolicy::getMaximumRedeliveries ( ) const`  
`[pure virtual]`

Gets the Maximum number of allowed redeliveries for a message before it will be discarded by the consumer.

#### Returns

maximum allowed redeliveries for a message.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1646).

6.665.3.7 `virtual long long activemq::core::RedeliveryPolicy::getRedeliveryDelay ( long long`  
`previousDelay ) [pure virtual]`

Given the last used redelivery delay calculate the next value of the delay based on the settings in this Policy instance.

#### Parameters

<i>previousDelay</i>	The last delay that was used between message redeliveries.
----------------------	--

#### Returns

the new delay to use before attempting another redelivery.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1646).

6.665.3.8 `virtual bool activemq::core::RedeliveryPolicy::isUseCollisionAvoidance ( ) const`  
`[pure virtual]`

#### Returns

whether or not collision avoidance is enabled for this Policy.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1646).

6.665.3.9 `virtual bool activemq::core::RedeliveryPolicy::isUseExponentialBackOff ( ) const`  
`[pure virtual]`

#### Returns

whether or not the exponential back off option is enabled.

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1647).

6.665.3.10 `virtual void activemq::core::RedeliveryPolicy::setBackOffMultiplier ( double value )`  
[pure virtual]

Sets the Back-Off Multiplier for Message Redelivery.

#### Parameters

<i>value</i>	The new value for the back-off multiplier.
--------------	--

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1647).

6.665.3.11 `virtual void activemq::core::RedeliveryPolicy::setCollisionAvoidancePercent ( short value )`  
[pure virtual]

#### Parameters

<i>value</i>	The collision avoidance percentage setting.
--------------	---

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1647).

6.665.3.12 `virtual void activemq::core::RedeliveryPolicy::setInitialRedeliveryDelay ( long long value )`  
[pure virtual]

Sets the initial time that redelivery will be delayed.

#### Parameters

<i>value</i>	Time in Milliseconds to wait before starting redelivery.
--------------	--

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1647).

6.665.3.13 `virtual void activemq::core::RedeliveryPolicy::setMaximumRedeliveries ( int maximumRedeliveries )`  
[pure virtual]

Sets the Maximum allowable redeliveries for a Message.

#### Parameters

<i>maximum-Redeliveries</i>	The maximum number of times that a message will be redelivered.
-----------------------------	---

Implemented in **activemq::core::policies::DefaultRedeliveryPolicy** (p. 1647).

6.665.3.14 `virtual void activemq::core::RedeliveryPolicy::setUseCollisionAvoidance ( bool value )` [pure virtual]

#### Parameters

<i>value</i>	Enable or Disable collision avoidance for this Policy.
--------------	--

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1648).

6.665.3.15 `virtual void activemq::core::RedeliveryPolicy::setUseExponentialBackOff ( bool value )` [pure virtual]

#### Parameters

<i>value</i>	Enable or Disable the exponential back off multiplier option.
--------------	---

Implemented in `activemq::core::policies::DefaultRedeliveryPolicy` (p. 1648).

### 6.665.4 Field Documentation

6.665.4.1 `const long long activemq::core::RedeliveryPolicy::NO_MAXIMUM_REDELIVERIES` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/RedeliveryPolicy.h`

## 6.666 decaf::util::concurrent::locks::ReentrantLock Class Reference

A reentrant mutual exclusion **Lock** (p. 2336) with extended capabilities.

```
#include <src/main/decaf/util/concurrent/locks/ReentrantLock.h>
```

Inheritance diagram for `decaf::util::concurrent::locks::ReentrantLock`:

### Public Member Functions

- **ReentrantLock** ()
- virtual **~ReentrantLock** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock.*
- virtual void **lockInterruptibly** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock unless the current thread is interrupted.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires the lock only if it is not held by another thread at the time of invocation.*
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException )  
*Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Attempts to release this lock.*
- virtual **Condition** \* **newCondition** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException )  
*Returns a **Condition** (p. 1220) instance for use with this **Lock** (p. 2336) instance.*
- int **getHoldCount** () const  
*Queries the number of holds on this lock by the current thread.*
- bool **isHeldByCurrentThread** () const  
*Queries if this lock is held by the current thread.*
- bool **isLocked** () const  
*Queries if this lock is held by any thread.*
- bool **isFair** () const  
*Returns true if this lock has fairness set true.*
- std::string **toString** () const  
*Returns a string identifying this lock, as well as its lock state.*

### 6.666.1 Detailed Description

A reentrant mutual exclusion **Lock** (p. 2336) with extended capabilities.

A **ReentrantLock** (p. 3126) is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking lock will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock. This can be checked using methods **isHeldByCurrentThread**() (p. 3129), and **getHoldCount**() (p. 3128).

The constructor for this class accepts an optional fairness parameter. When set true, under contention, locks favor granting access to the longest-waiting thread. Otherwise this lock does not guarantee any particular access order. Programs using fair locks accessed by many threads may display lower overall throughput (i.e., are slower; often much slower) than those using the default setting, but have smaller variances in times to obtain locks and guarantee lack of starvation. Note however, that fairness of locks does not guarantee fairness of thread scheduling. Thus, one of many threads using a fair lock may obtain it multiple times in succession while other active threads are not progressing and not currently holding the lock. Also note that the untimed tryLock method does not honor the fairness setting. It will succeed if the lock is available even if other threads are waiting.

It is recommended practice to always immediately follow a call to lock with a try block, most typically in a before/after construction such as:

```
class X { private:
```

```
    ReentrantLock (p. 3126) lock; // ...
```

```
public:
```

```
void m() { lock.lock(); // block until condition holds
```

```
try { // ... method body } finally { lock.unlock() } } }
```

In addition to implementing the **Lock** (p. 2336) interface, this class defines methods `isLocked` and `getLockQueueLength`, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

### Since

1.0

## 6.666.2 Constructor & Destructor Documentation

6.666.2.1 `decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ( )`

6.666.2.2 `virtual decaf::util::concurrent::locks::ReentrantLock::~~ReentrantLock ( )`  
`[virtual]`

## 6.666.3 Member Function Documentation

6.666.3.1 `int decaf::util::concurrent::locks::ReentrantLock::getHoldCount ( ) const`

Queries the number of holds on this lock by the current thread.

A thread has a hold on a lock for each lock action that is not matched by an unlock action.

The hold count information is typically only used for testing and debugging purposes. For example, if a certain section of code should not be entered with the lock already held then we can assert that fact:

```
class X { private:
```

```
    ReentrantLock (p. 3126) lock; // ...
```

```
public:
```

```
void m() { assert( lock.getHoldCount() == 0 ); lock.lock(); try { // ... method body }
catch(...) { lock.unlock(); } } }
```

### Returns

the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread



6.666.3.2 `bool decaf::util::concurrent::locks::ReentrantLock::isFair ( ) const`

Returns true if this lock has fairness set true.

#### Returns

true if this lock has fairness set true

6.666.3.3 `bool decaf::util::concurrent::locks::ReentrantLock::isHeldByCurrentThread ( ) const`

Queries if this lock is held by the current thread.

This method is typically used for debugging and testing. For example, a method that should only be called while a lock is held can assert that this is the case:

```
class X { private: ReentrantLock (p. 3126) lock = new ReentrantLock() (p. 3128); // ...  
public: void m() { assert( lock.isHeldByCurrentThread() ); // ... method body } }
```

It can also be used to ensure that a reentrant lock is used in a non-reentrant manner, for example:

```
class X { private: ReentrantLock (p. 3126) lock = new ReentrantLock() (p. 3128); // ...  
public: void m() { assert !lock.isHeldByCurrentThread() (p. 3129); lock.lock(); try { // ...  
method body } finally { lock.unlock(); } } }
```

#### Returns

true if current thread holds this lock and false otherwise

6.666.3.4 `bool decaf::util::concurrent::locks::ReentrantLock::isLocked ( ) const`

Queries if this lock is held by any thread.

This method is designed for use in monitoring of the system state, not for synchronization control.

#### Returns

true if any thread holds this lock and false otherwise

6.666.3.5 `virtual void decaf::util::concurrent::locks::ReentrantLock::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Acquires the lock.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds the lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired, at which time the lock hold count is set to one.

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 2338).

```
6.666.3.6  virtual void decaf::util::concurrent::locks::ReentrantLock::lockInterruptibly
            ( ) throw ( decaf::lang::exceptions::RuntimeException,
                      decaf::lang::exceptions::InterruptedException ) [virtual]
```

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread.

If the lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).
-----------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 2338).

```
6.666.3.7 virtual Condition* decaf::util::concurrent::locks::ReentrantLock::newCondition
( ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Returns a **Condition** (p. 1220) instance for use with this **Lock** (p. 2336) instance.

The returned **Condition** (p. 1220) instance supports the same usages as do the **Mutex** (p. 2736) Class' methods (wait, notify, and notifyAll).

\* If this lock is not held when any of the **Condition** (p. 1220) waiting or signalling methods are called, then an `IllegalMonitorStateException` is thrown. \* When the condition waiting methods are called the lock is released and, before they return, the lock is reacquired and the lock hold count restored to what it was when the method was called. \* If a thread is interrupted while waiting then the wait will terminate, an `InterruptedException` will be thrown, and the thread's interrupted status will be cleared. \* Waiting threads are signaled in FIFO order. \* The ordering of lock reacquisition for threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for fair locks favors those threads that have been waiting the longest.

### Exceptions

<i>RuntimeException</i>	if an error occurs while creating the <b>Condition</b> (p. 1220).
<i>UnsupportedOperationException</i>	if this <b>Lock</b> (p. 2336) implementation does not support conditions

Implements **decaf::util::concurrent::locks::Lock** (p. 2339).

```
6.666.3.8 std::string decaf::util::concurrent::locks::ReentrantLock::toString ( ) const
```

Returns a string identifying this lock, as well as its lock state.

The state, in brackets, includes either the String "Unlocked" or the String "Locked by" followed by the name of the owning thread.

### Returns

a string identifying this lock, as well as its lock state

```
6.666.3.9 virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ( long long time,
const TimeUnit & unit ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::InterruptedException ) [virtual]
```

Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock will not be acquired if any other threads are waiting

for the lock. This is in contrast to the **tryLock()** (p.3133) method. If you want a timed tryLock that does permit barging on a fair lock then combine the timed and un-timed forms together:

```
if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }
```

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* The lock is acquired by the current thread; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses

If the lock is acquired then the value true is returned and the lock hold count is set to one.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

#### Parameters

<i>time</i>	the maximum time to wait for the lock
<i>unit</i>	the time unit of the time argument

#### Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

#### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
<i>InterruptedException</i>	if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implements **decaf::util::concurrent::locks::Lock** (p.2339).

```
6.666.3.10 virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ( ) throw (
decaf::lang::exceptions::RuntimeException ) [virtual]
```

Acquires the lock only if it is not held by another thread at the time of invocation.

Acquires the lock if it is not held by another thread and returns immediately with the

## 6.667 decaf::util::concurrent::RejectedExecutionException Class Reference 3143

value true, setting the lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to **tryLock()** (p. 3133) will immediately acquire the lock if it is available, whether or not other threads are currently waiting for the lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting for this lock, then use **tryLock(0, TimeUnit.SECONDS)** (p. 3757)) which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then this method will return immediately with the value false.

### Returns

true if the lock was acquired and false otherwise

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 2340).

```
6.666.3.11  virtual void decaf::util::concurrent::locks::ReentrantLock::unlock
            ( ) throw ( decaf::lang::exceptions::RuntimeException,
                      decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]
```

Attempts to release this lock.

If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then **IllegalMonitorStateException** is thrown.

### Exceptions

<i>RuntimeException</i>	if an error occurs while acquiring the lock.
-------------------------	--

Implements **decaf::util::concurrent::locks::Lock** (p. 2341).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**ReentrantLock.h**

## 6.667 decaf::util::concurrent::RejectedExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/RejectedExecutionException.h>
```

Inheritance diagram for `decaf::util::concurrent::RejectedExecutionException`:

## Public Member Functions

- **RejectedExecutionException** () throw ()  
*Default Constructor.*
- **RejectedExecutionException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()  
*Copy Constructor.*
- **RejectedExecutionException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **RejectedExecutionException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **RejectedExecutionException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **RejectedExecutionException** \* **clone** () const  
*Clones this exception.*
- virtual ~**RejectedExecutionException** () throw ()

## 6.667.1 Constructor & Destructor Documentation

6.667.1.1 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( ) throw () [inline]`

Default Constructor.

6.667.1.2 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

### Parameters

<code>ex</code>	An exception that should become this type of Exception
-----------------	--

## 6.667 decaf::util::concurrent::RejectedExecutionException Class Reference 3145

6.667.1.3 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const RejectedExecutionException & ex ) throw () [inline]

Copy Constructor.

### Parameters

<i>ex</i>	- The Exception to copy in this new instance.
-----------	---

6.667.1.4 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const std::exception \* cause ) throw () [inline]

Constructor.

### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.667.1.5 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const char \* file, const int lineNumber, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.
<i>msg</i>	- The message to report
<i>...</i>	- list of primitives that are formatted into the message

6.667.1.6 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException ( const char \* file, const int lineNumber, const std::exception \* cause, const char \* msg, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	- The file name where exception occurs
<i>lineNumber</i>	- The line number where the exception occurred.

<i>cause</i>	- The exception that was the cause for this one to be thrown.
<i>msg</i>	- The message to report
...	- list of primitives that are formatted into the message

6.667.1.7 `virtual decaf::util::concurrent::RejectedExecutionException::~~RejectedExecutionException ( ) throw ( ) [inline, virtual]`

## 6.667.2 Member Function Documentation

6.667.2.1 `virtual RejectedExecutionException* decaf::util::concurrent::RejectedExecutionException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

A new instance this exception type with a copy the current state.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionException.h`

## 6.668 decaf::util::concurrent::RejectedExecutionHandler Class Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

```
#include <src/main/decaf/util/concurrent/RejectedExecutionHandler.h>
```

### Public Member Functions

- `virtual ~RejectedExecutionHandler ( )`
- `virtual void rejectedExecution (Runnable *r, ThreadPoolExecutor *executor)=0 throw ( RejectedExecutionException )`

*Method that may be invoked by a **ThreadPoolExecutor** (p. ??) when **execute** (p. ??) cannot accept a task.*



### 6.668.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

Since

1.0

### 6.668.2 Constructor & Destructor Documentation

6.668.2.1 `virtual decaf::util::concurrent::RejectedExecutionHandler::~RejectedExecutionHandler  
( ) [inline, virtual]`

### 6.668.3 Member Function Documentation

6.668.3.1 `virtual void decaf::util::concurrent::RejectedExecutionHandler::rejectedExecution  
( Runnable * r, ThreadPoolExecutor * executor ) throw ( RejectedExecutionException ) [pure virtual]`

Method that may be invoked by a **ThreadPoolExecutor** (p. ??) when **execute** (p. ??) cannot accept a task.

This may occur when no more threads or queue slots are available because their bounds would be exceeded, or upon shutdown of the **Executor** (p. 1831).

In the absence of other alternatives, the method may throw an **RejectedExecutionException** (p. 3134), which will be propagated to the caller of **execute** (p. ??).

#### Parameters

<i>r</i>	The pointer to the runnable task requested to be executed.
<i>executor</i>	The pointer to the executor attempting to execute this task.

#### Exceptions

<b>RejectedExecutionException</b> (p. 3134)	if there is no remedy.
--	------------------------

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionHandler.h`

## 6.669 activemq::commands::RemoveInfo Class Reference

```
#include <src/main/activemq/commands/RemoveInfo.h>
```

Inheritance diagram for `activemq::commands::RemoveInfo`:

## Public Member Functions

- **RemoveInfo** ()
- virtual **~RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **RemoveInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > & **getObjectId** () const
- virtual **Pointer**< **DataStructure** > & **getObjectId** ()
- virtual void **setObjectId** (const **Pointer**< **DataStructure** > &objectId)
- virtual long long **getLastDeliveredSequenceld** () const
- virtual void **setLastDeliveredSequenceld** (long long lastDeliveredSequenceld)
- virtual bool **isRemoveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

## Static Public Attributes

- static const unsigned char **ID\_REMOVEINFO** = 12

## Protected Attributes

- **Pointer**< **DataStructure** > **objectId**
- long long **lastDeliveredSequenceld**

## 6.669.1 Constructor & Destructor Documentation

6.669.1.1 **activemq::commands::RemoveInfo::RemoveInfo** ( )

6.669.1.2 **virtual activemq::commands::RemoveInfo::~~RemoveInfo** ( ) [virtual]

## 6.669.2 Member Function Documentation

6.669.2.1 virtual **RemoveInfo\*** **activemq::commands::RemoveInfo::cloneDataStructure** ( )  
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.669.2.2 virtual void **activemq::commands::RemoveInfo::copyDataStructure** ( const **DataStructure \* src** ) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.669.2.3 virtual bool **activemq::commands::RemoveInfo::equals** ( const **DataStructure \* value** ) const [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.669.2.4 virtual unsigned char **activemq::commands::RemoveInfo::getDataStructureType** ( )  
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.669.2.5 `virtual long long activemq::commands::RemoveInfo::getLastDeliveredSequenceld ( ) const [virtual]`

6.669.2.6 `virtual const Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId ( ) const [virtual]`

6.669.2.7 `virtual Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId ( ) [virtual]`

6.669.2.8 `virtual bool activemq::commands::RemoveInfo::isRemoveInfo ( ) const [inline, virtual]`

### Returns

an answer of true to the **isRemoveInfo()** (p. 3140) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 728).

6.669.2.9 `virtual void activemq::commands::RemoveInfo::setLastDeliveredSequenceld ( long long lastDeliveredSequenceld ) [virtual]`

6.669.2.10 `virtual void activemq::commands::RemoveInfo::setObjectId ( const Pointer<DataStructure> & objectId ) [virtual]`

6.669.2.11 `virtual std::string activemq::commands::RemoveInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.669.2.12 `virtual Pointer<Command> activemq::commands::RemoveInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

## Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

## 6.669.3 Field Documentation

6.669.3.1 `const unsigned char activemq::commands::RemoveInfo::ID_REMOVEINFO = 12` [static]

6.669.3.2 `long long activemq::commands::RemoveInfo::lastDeliveredSequenceId` [protected]

6.669.3.3 `Pointer<DataStructure> activemq::commands::RemoveInfo::objectId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveInfo.h`

## 6.670 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3141).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller`:

## Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataInputStream \* dataIn**, **utils::BooleanStream \* bs**)  
**throw** ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.670.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3141).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.670.2 Constructor & Destructor Documentation

- 6.670.2.1 **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]
- 6.670.2.2 **virtual activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.670.3 Member Function Documentation

- 6.670.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.670.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.670.3.3 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.670.3.4 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.670.3.5  virtual int activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.670.3.6  virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).



## 6.671 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller

### Class Reference 3155

```
6.670.3.7 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveInfoMarshaller.h**

## 6.671 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3145).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller**:

#### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataInputStream \* dataIn**, **utils::BooleanStream \* bs**)  
**throw ( decaf::io::IOException )**

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.671.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3145).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.671.2 Constructor & Destructor Documentation

6.671.2.1 **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]

6.671.2.2 **virtual activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.671.3 Member Function Documentation

6.671.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.671.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.671.3.3 virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.671.3.4 virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.671.3.5  virtual int activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.671.3.6  virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

## 6.672 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller

### Class Reference 3159

6.671.3.7 virtual void activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**RemoveInfoMarshaller.h**

## 6.672 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3149).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller:

#### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.672.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3149).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.672.2 Constructor & Destructor Documentation

6.672.2.1 **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]

6.672.2.2 **virtual activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.672.3 Member Function Documentation

6.672.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.672.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.672.3.3 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.672.3.4 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.672.3.5  virtual int activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.672.3.6  virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).



## 6.673 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller

### Class Reference 3163

6.672.3.7 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveInfoMarshaller.h**

## 6.673 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3153).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller:

#### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.673.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3153).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.673.2 Constructor & Destructor Documentation

6.673.2.1 **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]

6.673.2.2 **virtual activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.673.3 Member Function Documentation

6.673.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.673.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.673.3.3 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.673.3.4 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.673.3.5  virtual int activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.673.3.6  virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

## 6.674 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller

### Class Reference 3167

6.673.3.7 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveInfoMarshaller.h**

## 6.674 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3157).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller:

#### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.674.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p.3157).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.674.2 Constructor & Destructor Documentation

6.674.2.1 **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]

6.674.2.2 **virtual activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.674.3 Member Function Documentation

6.674.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1578).

6.674.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaller.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.674.3.3 virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.674.3.4 virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.674.3.5  virtual int activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.674.3.6  virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).



## 6.675 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller

### Class Reference 3171

6.674.3.7 virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**RemoveInfoMarshaller.h**

## 6.675 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3161).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller:

#### Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.675.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p.3161).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.675.2 Constructor & Destructor Documentation

6.675.2.1 **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::RemoveInfoMarshaller**  
( ) [inline]

6.675.2.2 **virtual activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::~~RemoveInfoMarshaller**  
( ) [inline, virtual]

### 6.675.3 Member Function Documentation

6.675.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1578).

6.675.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.675.3.3 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.675.3.4 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.675.3.5  virtual int activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.675.3.6  virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.675.3.7 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**RemoveInfoMarshaller.h**

## 6.676 activemq::commands::RemoveSubscriptionInfo Class Reference

```
#include <src/main/activemq/commands/RemoveSubscriptionInfo.h>
```

Inheritance diagram for activemq::commands::RemoveSubscriptionInfo:

#### Public Member Functions

- **RemoveSubscriptionInfo** ()
- virtual **~RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **RemoveSubscriptionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*

- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_REMOVESUBSCRIPTIONINFO** = 9

### Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **subscriptionName**
- std::string **clientId**

## 6.676.1 Constructor & Destructor Documentation

6.676.1.1 **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** ( )

6.676.1.2 **virtual activemq::commands::RemoveSubscriptionInfo::~~RemoveSubscriptionInfo** ( ) [virtual]

## 6.676.2 Member Function Documentation

6.676.2.1 **virtual RemoveSubscriptionInfo\* activemq::commands::RemoveSubscriptionInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

**6.676.2.2** `virtual void activemq::commands::RemoveSubscriptionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

**6.676.2.3** `virtual bool activemq::commands::RemoveSubscriptionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

**6.676.2.4** `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId ( ) const [virtual]`

**6.676.2.5** `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId ( ) [virtual]`

**6.676.2.6** `virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId ( ) const [virtual]`

**6.676.2.7** `virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId ( ) [virtual]`

6.676.2.8 `virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.676.2.9 `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName ( ) const [virtual]`

6.676.2.10 `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName ( ) [virtual]`

6.676.2.11 `virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isRemoveSubscriptionInfo()** (p. 3168) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 728).

6.676.2.12 `virtual void activemq::commands::RemoveSubscriptionInfo::setClientId ( const std::string & clientId ) [virtual]`

6.676.2.13 `virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId ( const Pointer< ConnectionId > & connectionId ) [virtual]`

6.676.2.14 `virtual void activemq::commands::RemoveSubscriptionInfo::setSubscriptionName ( const std::string & subscriptionName ) [virtual]`

6.676.2.15 `virtual std::string activemq::commands::RemoveSubscriptionInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).



## 6.677 ac-

### activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller Class Reference 3179

```
6.676.2.16 virtual Pointer<Command> ac-  
    tivemq::commands::RemoveSubscriptionInfo::visit (  
        activemq::state::CommandVisitor * visitor ) throw (  
            exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

#### 6.676.3 Field Documentation

6.676.3.1 **std::string activemq::commands::RemoveSubscriptionInfo::clientId**  
[protected]

6.676.3.2 **Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId**  
[protected]

6.676.3.3 **const unsigned char activemq::commands::RemoveSubscriptionInfo::ID\_-  
REMOVESUBSCRIPTIONINFO = 9** [static]

6.676.3.4 **std::string activemq::commands::RemoveSubscriptionInfo::subscriptionName**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**RemoveSubscriptionInfo.h**

## 6.677 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3169).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMa
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller:

## Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.677.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3169).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.677.2 Constructor & Destructor Documentation

- 6.677.2.1 **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller** ( ) [*inline*]
- 6.677.2.2 **virtual activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller** ( ) [*inline, virtual*]

### 6.677.3 Member Function Documentation

6.677 ac-

**ativemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

**Class Reference**

**3181**

```
6.677.3.1 virtual commands::DataStructure* ac-  
ativemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::createObject  
( ) const [virtual]
```

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.677.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::getDataStructureType  
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.677.3.3 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseMarshal  
( OpenWireFormat * wireFormat, commands::DataStructure *  
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (  
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**  
(p. 745).

```
6.677.3.4  virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
    [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.677.3.5  virtual int activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
    [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

6.677 ac-

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

3183

```
6.677.3.6 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.677.3.7 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveSubscriptionInfoMarshaller.h**

## 6.678 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3174).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscript
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.678.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3174).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.678 ac-

tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

Class Reference

3185

## 6.678.2 Constructor & Destructor Documentation

6.678.2.1 `ativemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller ( ) [inline]`

6.678.2.2 `virtual ativemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller ( ) [inline, virtual]`

## 6.678.3 Member Function Documentation

6.678.3.1 `virtual commands::DataStructure* ativemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.678.3.2 `virtual unsigned char ativemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **ativemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.678.3.3 `virtual void ativemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.678.3.4  virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.678.3.5  virtual int activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



## 6.678 ac-

**activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

### Class Reference

3187

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.678.3.6 virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.678.3.7 virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveSubscriptionInfoMarshaller.h**

## 6.679 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3178).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscript
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.679.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3178).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.679.2.1 `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller`  
`( ) [inline]`

6.679.2.2 `virtual activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller`  
`( ) [inline, virtual]`

### 6.679.3 Member Function Documentation

6.679.3.1 `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::createObject`  
`( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.679.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::getDataStructureType`  
`( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.679.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseMarshal`  
`( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.679.3.4  virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.679.3.5  virtual int activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.679 ac-

**activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

### Class Reference

3191

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.679.3.6 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.679.3.7 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveSubscriptionInfoMarshaller.h**

## 6.680 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3182).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscript
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.680.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3182).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.680.2.1 `activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller`  
`( ) [inline]`

6.680.2.2 `virtual activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller`  
`( ) [inline, virtual]`

### 6.680.3 Member Function Documentation

6.680.3.1 `virtual commands::DataStructure* ac-`  
`tivemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::createObject`  
`( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.680.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::getDataStructureType`  
`( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.680.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::looseMarshal`  
`( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

```
6.680.3.4  virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.680.3.5  virtual int activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



6.680 ac-

ativemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller

Class Reference

3195

Reimplemented from **ativemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.680.3.6  virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.680.3.7  virtual void activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
              bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/ativemq/wireformat/openwire/marshal/v6/**RemoveSubscriptionInfoMarshaller.h**

## 6.681 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3186).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscript
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.681.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3186).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.681.2.1 `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller ( ) [inline]`

6.681.2.2 `virtual activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller ( ) [inline, virtual]`

### 6.681.3 Member Function Documentation

6.681.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.681.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.681.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.681.3.4  virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

```
6.681.3.5  virtual int activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

6.681 ac-

ativemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

**Class Reference**

3199

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.681.3.6 virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.681.3.7 virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/ativemq/wireformat/openwire/marshal/v5/**RemoveSubscriptionInfoMarshaller.h**

## 6.682 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3190).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscript
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller:

### Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.682.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3190).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.682.3.4  virtual void activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.682.3.5  virtual int activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



6.682 ac-

ativemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

Class Reference

3203

Reimplemented from **ativemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.682.3.6 virtual void ativemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.682.3.7 virtual void ativemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **ativemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/ativemq/wireformat/openwire/marshal/v4/**RemoveSubscriptionInfoMarshaller.h**

## 6.683 activemq::commands::ReplayCommand Class Reference

```
#include <src/main/activemq/commands/ReplayCommand.h>
```

Inheritance diagram for activemq::commands::ReplayCommand:

### Public Member Functions

- **ReplayCommand** ()
- virtual **~ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ReplayCommand** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int firstNakNumber)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int lastNakNumber)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_REPLAYCOMMAND** = 65

### Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

### 6.683.1 Constructor & Destructor Documentation

6.683.1.1 `activemq::commands::ReplayCommand::ReplayCommand ( )`

6.683.1.2 `virtual activemq::commands::ReplayCommand::~~ReplayCommand ( )`  
[virtual]

### 6.683.2 Member Function Documentation

6.683.2.1 `virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.683.2.2 `virtual void activemq::commands::ReplayCommand::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.683.2.3 `virtual bool activemq::commands::ReplayCommand::equals ( const DataStructure * value ) const` [virtual]

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.683.2.4 `virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.683.2.5 `virtual int activemq::commands::ReplayCommand::getFirstNakNumber ( ) const [virtual]`

6.683.2.6 `virtual int activemq::commands::ReplayCommand::getLastNakNumber ( ) const [virtual]`

6.683.2.7 `virtual void activemq::commands::ReplayCommand::setFirstNakNumber ( int firstNakNumber ) [virtual]`

6.683.2.8 `virtual void activemq::commands::ReplayCommand::setLastNakNumber ( int lastNakNumber ) [virtual]`

6.683.2.9 `virtual std::string activemq::commands::ReplayCommand::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.683.2.10 `virtual Pointer<Command> activemq::commands::ReplayCommand::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.683.3 Field Documentation

- 6.683.3.1 `int activemq::commands::ReplayCommand::firstNakNumber`  
[protected]
- 6.683.3.2 `const unsigned char activemq::commands::ReplayCommand::ID_-REPLAYCOMMAND = 65` [static]
- 6.683.3.3 `int activemq::commands::ReplayCommand::lastNakNumber`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ReplayCommand.h`

## 6.684 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3197).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.684.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3197).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.684.2 Constructor & Destructor Documentation

6.684.2.1 **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::ReplayCommandMarshaller**  
 ( ) [inline]

6.684.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
 ( ) [inline, virtual]

### 6.684.3 Member Function Documentation

6.684.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::createObject**  
 ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.684.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::getDataStructureType**  
 ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.684 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller

### Class Reference 3209

6.684.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.684.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

6.684.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.684.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.684.3.7  virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.685 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller

### Class Reference

3211

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ReplayCommandMarshaller.h**

## 6.685 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3201).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.685.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3201).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.685.2 Constructor & Destructor Documentation

6.685.2.1 **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.685.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.685.3 Member Function Documentation

6.685.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.685.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.685.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.685.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

6.685.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.685.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.685.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

## 6.686 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller

### Class Reference

3215

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ReplayCommandMarshaller.h**

## 6.686 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3205).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.686.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3205).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.686.2 Constructor & Destructor Documentation

6.686.2.1 **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.686.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.686.3 Member Function Documentation

6.686.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.686.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.686 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller

### Class Reference 3217

6.686.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.686.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

6.686.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.686.3.6  virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.686.3.7  virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.687 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller

### Class Reference

3219

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ReplayCommandMarshaller.h**

## 6.687 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3209).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual ~**ReplayCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.687.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3209).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.687.2 Constructor & Destructor Documentation

6.687.2.1 **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.687.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.687.3 Member Function Documentation

6.687.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.687.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.687 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller

### Class Reference 3221

6.687.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.687.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.687.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.687.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.687.3.7  virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

## 6.688 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller

### Class Reference

3223

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ReplayCommandMarshaller.h**

## 6.688 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3213).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller**:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.688.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3213).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.688.2 Constructor & Destructor Documentation

6.688.2.1 **activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.688.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.688.3 Member Function Documentation

6.688.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.688.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.688 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller

### Class Reference 3225

6.688.3.3 virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.688.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.688.3.5 virtual int activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.688.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.688.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.689 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller

### Class Reference

3227

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ReplayCommandMarshaller.h**

## 6.689 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3217).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**:

### Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual ~**ReplayCommandMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.689.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3217).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.689.2 Constructor & Destructor Documentation

6.689.2.1 **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::ReplayCommandMarshaller**  
( ) [inline]

6.689.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::~~ReplayCommandMarshaller**  
( ) [inline, virtual]

### 6.689.3 Member Function Documentation

6.689.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::createObject**  
( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.689.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::getDataStructureType**  
( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.689.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.689.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

6.689.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.689.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.689.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ReplayCommandMarshaller.h**

## 6.690 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveProducerExecutor:

### Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** \**action*, **CmsTemplate** \**parent*, const std::string &*destinationName*)
- virtual ~**ResolveProducerExecutor** ()
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \**session*) throw ( cms::CMSException )

### Protected Member Functions

- **ResolveProducerExecutor** & **operator=** (const **ResolveProducerExecutor** &)

### 6.690.1 Constructor & Destructor Documentation

6.690.1.1 **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor** ( **ProducerCallback** \* *action*, **CmsTemplate** \* *parent*, const std::string & *destinationName* ) [inline]

6.690.1.2 `virtual activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::~~ResolveProducerExecutor ( ) [inline, virtual]`

## 6.690.2 Member Function Documentation

6.690.2.1 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination ( cms::Session * session ) throw ( cms::CMSEException ) [virtual]`

6.690.2.2 `ResolveProducerExecutor& activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::operator= ( const ResolveProducerExecutor & ) [inline, protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

## 6.691 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor`:

### Public Member Functions

- **ResolveReceiveExecutor** (**CmsTemplate** \*parent, const std::string &selector, bool noLocal, const std::string &destinationName)
- virtual **~ResolveReceiveExecutor** ()
- virtual **cms::Destination** \* **getDestination** (**cms::Session** \*session) throw ( cms::CMSEException )

### Protected Member Functions

- **ResolveReceiveExecutor** & **operator=** (const **ResolveReceiveExecutor** &)

## 6.691.1 Constructor & Destructor Documentation

6.691.1.1 `activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor ( CmsTemplate * parent, const std::string & selector, bool noLocal, const std::string & destinationName ) [inline]`

6.691.1.2 virtual activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::~ResolveReceiveExecutor  
( ) [inline, virtual]

## 6.691.2 Member Function Documentation

6.691.2.1 virtual cms::Destination\* activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination  
( cms::Session \* session ) throw ( cms::CMSException ) [virtual]

6.691.2.2 ResolveReceiveExecutor& activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::operator= ( const  
ResolveReceiveExecutor & ) [inline, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

## 6.692 decaf::internal::util::Resource Class Reference

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

```
#include <src/main/decaf/internal/util/Resource.h>
```

Inheritance diagram for decaf::internal::util::Resource:

### Public Member Functions

- virtual ~**Resource** ()

### 6.692.1 Detailed Description

Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.

#### Since

1.0

### 6.692.2 Constructor & Destructor Documentation

6.692.2.1 virtual decaf::internal::util::Resource::~Resource ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**Resource.h**

## 6.693 decaf::internal::util::ResourceLifecycleManager Class Reference

```
#include <src/main/decaf/internal/util/ResourceLifecycleManager.h>
```

### Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
- virtual void **addResource** (**Resource** \*value)

### Protected Member Functions

- virtual void **destroyResources** ()

#### 6.693.1 Detailed Description

Since

1.0

#### 6.693.2 Constructor & Destructor Documentation

6.693.2.1 **decaf::internal::util::ResourceLifecycleManager::ResourceLifecycleManager** ( )

6.693.2.2 virtual **decaf::internal::util::ResourceLifecycleManager::~~ResourceLifecycleManager** ( ) [virtual]

#### 6.693.3 Member Function Documentation

6.693.3.1 virtual void **decaf::internal::util::ResourceLifecycleManager::addResource** ( **Resource** \* *value* ) [virtual]

6.693.3.2 virtual void **decaf::internal::util::ResourceLifecycleManager::destroyResources** ( ) [protected, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ResourceLifecycleManager.h**

## 6.694 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

```
#include <src/main/activemq/cmsutil/ResourceLifecycleManager.h>
```



## Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()  
*Destructor - calls `destroy`*
- void **addConnection** (**cms::Connection** \*connection) throw ( cms::CMSException )  
*Adds a connection so that its life will be managed by this object.*
- void **addSession** (**cms::Session** \*session) throw ( cms::CMSException )  
*Adds a session so that its life will be managed by this object.*
- void **addDestination** (**cms::Destination** \*dest) throw ( cms::CMSException )  
*Adds a destination so that its life will be managed by this object.*
- void **addMessageProducer** (**cms::MessageProducer** \*producer) throw ( cms::CMSException )  
*Adds a message producer so that its life will be managed by this object.*
- void **addMessageConsumer** (**cms::MessageConsumer** \*consumer) throw ( cms::CMSException )  
*Adds a message consumer so that its life will be managed by this object.*
- void **destroy** () throw ( cms::CMSException )  
*Closes and destroys the contained CMS resources.*
- void **releaseAll** ()  
*Releases all of the contained resources so that this object will no longer control their lifetimes.*

## Protected Member Functions

- **ResourceLifecycleManager** (const **ResourceLifecycleManager** &)
- **ResourceLifecycleManager** & **operator=** (const **ResourceLifecycleManager** &)

### 6.694.1 Detailed Description

Manages the lifecycle of a set of CMS resources.

A call to `destroy` will close and destroy all of the contained resources in the appropriate manner.

### 6.694.2 Constructor & Destructor Documentation

6.694.2.1 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager** ( const **ResourceLifecycleManager** & ) [*inline, protected*]

6.694.2.2 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager** ( )

6.694.2.3 `virtual activemq::cmsutil::ResourceLifecycleManager::~~ResourceLifecycleManager ( ) [virtual]`

Destructor - calls `destroy`

### 6.694.3 Member Function Documentation

6.694.3.1 `void activemq::cmsutil::ResourceLifecycleManager::addConnection ( cms::Connection * connection ) throw ( cms::CMSEException )`

Adds a connection so that its life will be managed by this object.

#### Parameters

<i>connection</i>	the object to be managed
-------------------	--------------------------

6.694.3.2 `void activemq::cmsutil::ResourceLifecycleManager::addDestination ( cms::Destination * dest ) throw ( cms::CMSEException )`

Adds a destination so that its life will be managed by this object.

#### Parameters

<i>dest</i>	the object to be managed
-------------	--------------------------

6.694.3.3 `void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer ( cms::MessageConsumer * consumer ) throw ( cms::CMSEException )`

Adds a message consumer so that its life will be managed by this object.

#### Parameters

<i>consumer</i>	the object to be managed
-----------------	--------------------------

6.694.3.4 `void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer ( cms::MessageProducer * producer ) throw ( cms::CMSEException )`

Adds a message producer so that its life will be managed by this object.

#### Parameters

<i>producer</i>	the object to be managed
-----------------	--------------------------

6.694.3.5 `void activemq::cmsutil::ResourceLifecycleManager::addSession ( cms::Session * session ) throw ( cms::CMSEException )`

Adds a session so that its life will be managed by this object.

#### Parameters

<i>session</i>	the object to be managed
----------------	--------------------------

6.694.3.6 `void activemq::cmsutil::ResourceLifecycleManager::destroy ( ) throw ( cms::CMSEException )`

Closes and destroys the contained CMS resources.

#### Exceptions

<b><i>cms::CMSEException</i></b> (p. 1130)	thrown if an error occurs.
---	----------------------------

6.694.3.7 `ResourceLifecycleManager& activemq::cmsutil::ResourceLifecycleManager::operator= ( const ResourceLifecycleManager & ) [inline, protected]`

6.694.3.8 `void activemq::cmsutil::ResourceLifecycleManager::releaseAll ( )`

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/ResourceLifecycleManager.h`

## 6.695 activemq::commands::Response Class Reference

```
#include <src/main/activemq/commands/Response.h>
```

Inheritance diagram for `activemq::commands::Response`:

#### Public Member Functions

- `Response ()`
- `virtual ~Response ()`

- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **Response** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual int **getCorrelationId** () const
- virtual void **setCorrelationId** (int correlationId)
- virtual bool **isResponse** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_RESPONSE** = 30

### Protected Attributes

- int **correlationId**

## 6.695.1 Constructor & Destructor Documentation

6.695.1.1 **activemq::commands::Response::Response** ( )

6.695.1.2 **virtual activemq::commands::Response::~~Response** ( ) [virtual]

## 6.695.2 Member Function Documentation

6.695.2.1 **virtual Response\*** **activemq::commands::Response::cloneDataStructure** ( ) const  
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1494), **activemq::commands::DataResponse** (p. 1551), **activemq::commands::ExceptionResponse** (p. 1803), and **activemq::commands::IntegerResponse** (p. 2055).

**6.695.2.2** `virtual void activemq::commands::Response::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

**Parameters**

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1494), **activemq::commands::DataResponse** (p. 1551), **activemq::commands::ExceptionResponse** (p. 1803), and **activemq::commands::IntegerResponse** (p. 2055).

**6.695.2.3** `virtual bool activemq::commands::Response::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse** (p. 1551), **activemq::commands::ExceptionResponse** (p. 1803), and **activemq::commands::IntegerResponse** (p. 2055).

**6.695.2.4** `virtual int activemq::commands::Response::getCorrelationId ( ) const [virtual]`

```
6.695.2.5 virtual unsigned char activemq::commands::Response::getDataStructureType ( )
          const [virtual]
```

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSet** (p. 1628) type copy.

Implements **activemq::commands::DataSet** (p. 1631).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse** (p. 1552), **activemq::commands::ExceptionResponse** (p. 1803), and **activemq::commands::IntegerResponse** (p. 2056).

```
6.695.2.6 virtual bool activemq::commands::Response::isResponse ( ) const [inline,
          virtual]
```

#### Returns

an answer of true to the **isResponse()** (p. 3230) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 728).

```
6.695.2.7 virtual void activemq::commands::Response::setCorrelationId ( int correlationId )
          [virtual]
```

```
6.695.2.8 virtual std::string activemq::commands::Response::toString ( ) const
          [virtual]
```

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1495), **activemq::commands::DataResponse** (p. 1552), **activemq::commands::ExceptionResponse** (p. 1804), and **activemq::commands::IntegerResponse** (p. 2056).

```
6.695.2.9 virtual Pointer<Command> activemq::commands::Response::visit
          ( activemq::state::CommandVisitor * visitor ) throw (
            exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

**Returns**

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

**6.695.3 Field Documentation**

6.695.3.1 **int** **activemq::commands::Response::correlationId** [protected]

6.695.3.2 **const unsigned char** **activemq::commands::Response::ID\_RESPONSE** = 30  
[static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Response.h**

**6.696 activemq::transport::mock::ResponseBuilder Class Reference**

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

```
#include <src/main/activemq/transport/mock/ResponseBuilder.h>
```

Inheritance diagram for **activemq::transport::mock::ResponseBuilder**:

**Public Member Functions**

- virtual **~ResponseBuilder** ()
- virtual **Pointer< Response > buildResponse** (const **Pointer< Command > &command**)=0

*Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.*

- virtual void **buildIncomingCommands** (const **Pointer< Command > &command**, **decaf::util::StlQueue< Pointer< Command > > &queue**)=0

*When called the **ResponseBuilder** (p. 3231) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

**6.696.1 Detailed Description**

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

## 6.696.2 Constructor & Destructor Documentation

6.696.2.1 `virtual activemq::transport::mock::ResponseBuilder::~~ResponseBuilder ( )`  
`[inline, virtual]`

## 6.696.3 Member Function Documentation

6.696.3.1 `virtual void activemq::transport::mock::ResponseBuilder::buildIncomingCommands (`  
`const Pointer< Command > & command, decaf::util::StlQueue< Pointer<`  
`Command >> & queue )` `[pure virtual]`

When called the **ResponseBuilder** (p. 3231) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

### Parameters

<i>command</i>	- The Command being sent to the Broker.
<i>queue</i>	- Queue of Command sent back from the broker.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2855).

6.696.3.2 `virtual Pointer<Response> activemq::transport::mock::ResponseBuilder::buildResponse (`  
`const Pointer< Command > & command )` `[pure virtual]`

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

### Parameters

<i>command</i>	- The command to build a response for
----------------	---------------------------------------

### Returns

A Response object pointer, or NULL if no response.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2856).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/ResponseBuilder.h`

## 6.697 **activemq::transport::correlator::ResponseCorrelator** Class Reference

This type of transport filter is responsible for correlating asynchronous responses with requests.



## 6.697 activemq::transport::correlator::ResponseCorrelator Class Reference 3243

```
#include <src/main/activemq/transport/correlator/ResponseCorrelator.h>
```

Inheritance diagram for activemq::transport::correlator::ResponseCorrelator:

### Public Member Functions

- **ResponseCorrelator** (const **Pointer**< **Transport** > &next)  
*Constructor.*
- virtual ~**ResponseCorrelator** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given request to the server and waits for the response.*
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*This is called in the context of the nested transport's reading thread.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual void **onTransportException** (**Transport** \*source, const decaf::lang::Exception &ex)  
*Event handler for an exception from a command transport.*

### 6.697.1 Detailed Description

This type of transport filter is responsible for correlating asynchronous responses with requests.

Non-response messages are simply sent directly to the CommandListener. It owns the transport that it

### 6.697.2 Constructor & Destructor Documentation

6.697.2.1 **activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator** ( const **Pointer**< **Transport** > & next )

Constructor.

**Parameters**

<i>next</i>	the next transport in the chain
-------------	---------------------------------

6.697.2.2 `virtual activemq::transport::correlator::ResponseCorrelator::~~ResponseCorrelator ( ) [virtual]`

**6.697.3 Member Function Documentation**

6.697.3.1 `virtual void activemq::transport::correlator::ResponseCorrelator::close ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

**Exceptions**

<i>IOException</i>	if errors occur.
--------------------	------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3829).

6.697.3.2 `virtual void activemq::transport::correlator::ResponseCorrelator::onCommand ( const Pointer< Command > & command ) [virtual]`

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

**Parameters**

<i>command</i>	the received from the nested transport.
----------------	---

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

6.697.3.3 `virtual void activemq::transport::correlator::ResponseCorrelator::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

**Parameters**

<i>command</i>	the command to be sent.
----------------	-------------------------

## 6.697 activemq::transport::correlator::ResponseCorrelator Class Reference 3245

### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3832).

6.697.3.4 `virtual void activemq::transport::correlator::ResponseCorrelator::onTransportException ( Transport * source, const decaf::lang::Exception & ex ) [virtual]`

Event handler for an exception from a command transport.

### Parameters

<i>source</i>	The source of the exception
<i>ex</i>	The exception.

6.697.3.5 `virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends the given request to the server and waits for the response.

### Parameters

<i>command</i>	The request to send.
----------------	----------------------

### Returns

the response from the server.

### Exceptions

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3833).

6.697.3.6 `virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request ( const Pointer< Command > & command, unsigned int timeout ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [virtual]`

Sends the given request to the server and waits for the response.

**Parameters**

<i>command</i>	The request to send.
<i>timeout</i>	The time to wait for a response.

**Returns**

the response from the server.

**Exceptions**

<i>IOException</i>	if an error occurs with the request.
--------------------	--------------------------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3833).

```
6.697.3.7 virtual void activemq::transport::correlator::ResponseCorrelator::start ( ) throw (
    decaf::io::IOException ) [virtual]
```

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

**Exceptions**

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Reimplemented from **activemq::transport::TransportFilter** (p. 3834).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

## 6.698 **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3236).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshal
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**:

**Public Member Functions**

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()

- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.698.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3236).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.698.2 Constructor & Destructor Documentation

- 6.698.2.1 **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::ResponseMarshaller**  
( ) [inline]
- 6.698.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::~~ResponseMarshaller**  
( ) [inline, virtual]

### 6.698.3 Member Function Documentation

- 6.698.3.1 **virtual commands::DataStructure\*** **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::createObject** ( )  
const [virtual]

Creates a new instance of this marshalable type.

**Returns**

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1505), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1570), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1822), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2070).

**6.698.3.2** `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1506), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1571), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1822), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2070).

**6.698.3.3** `virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

## 6.698 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference 3249

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1506), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1571), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1822), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2070).

```
6.698.3.4  virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1506), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1571), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1823), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2071).

```
6.698.3.5  virtual int activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1507), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1572), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1823), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2071).

```
6.698.3.6  virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1507), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1572), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1824), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2072).

```
6.698.3.7  virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.699 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference 3251

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1508), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1573), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1824), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 2072).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ResponseMarshaller.h**

## 6.699 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3241).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.699.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3241).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.699.2 Constructor & Destructor Documentation

6.699.2.1 **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::ResponseMarshaller**  
 ( ) [inline]

6.699.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::~~ResponseMarshaller**  
 ( ) [inline, virtual]

### 6.699.3 Member Function Documentation

6.699.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1497), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**

## 6.699 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference 3253

(p. 1562), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1810), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2062).

6.699.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1497), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1562), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1810), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2062).

6.699.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1498), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1563), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1810), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2062).

```
6.699.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1498), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1563), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1811), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2063).

```
6.699.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

## 6.699 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference 3255

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1498), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1564), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1811), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2063).

```
6.699.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1499), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1564), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1812), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2064).

```
6.699.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i> if an error occurs during the unmarshal.
---

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1499), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1565), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1812), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 2064).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ResponseMarshaller.h**

## 6.700 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3246).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshal
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**:

## Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.700.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3246).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.700.2 Constructor & Destructor Documentation

6.700.2.1 **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::ResponseMarshaller**  
( ) [inline]

6.700.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::~~ResponseMarshaller**  
( ) [inline, virtual]

### 6.700.3 Member Function Documentation

6.700.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1513), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1554), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1818), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2078).

6.700.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1514), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1554), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1818), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2078).

```
6.700.3.3  virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1514), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1554), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1818), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2078).

```
6.700.3.4  virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled



## 6.700 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference 3259

<i>dataIn</i>	- BinaryReader that provides that data source
---------------	---

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1514), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1555), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1819), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2079).

```
6.700.3.5  virtual int activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal1
           ( OpenWireFormat * wireFormat, commands::DataStructure *
             dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
           [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1515), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1555), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1819), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2079).

```
6.700.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1515), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1556), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1820), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2080).

```
6.700.3.7 virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference 3261

Reimplemented in **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller** (p. 1516), **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller** (p. 1556), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1820), and **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller** (p. 2080).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ResponseMarshaller.h**

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3250).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.701.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3250).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.701.2 Constructor & Destructor Documentation

6.701.2.1 **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::ResponseMarshaller**  
( ) [inline]

6.701.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::~~ResponseMarshaller**  
( ) [inline, virtual]

### 6.701.3 Member Function Documentation

6.701.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1501), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1566), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1814), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2066).

6.701.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference 3263

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1501), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1567), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1814), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2066).

```
6.701.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1502), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1567), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1814), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2066).

```
6.701.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1502), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1567), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1815), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2067).

```
6.701.3.5  virtual int activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1503), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1568), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1815), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2067).

```
6.701.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

## 6.701 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference 3265

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1503), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1568), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1816), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2068).

```
6.701.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1504), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1569), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1816), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 2068).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ResponseMarshaller.h**

## 6.702 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3255).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshal
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ResponseMarshaller:

### Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.702.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3255).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.702.2 Constructor & Destructor Documentation

6.702.2.1 `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::ResponseMarshaller`  
`( ) [inline]`

6.702.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::~~ResponseMarshaller`  
`( ) [inline, virtual]`

## 6.702.3 Member Function Documentation

6.702.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::createObject ( )`  
`const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1509), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1574), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1826), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2074).

6.702.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::getDataStructureType`  
`( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1510), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1575), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1826), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 2074).

```
6.702.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1510), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1575), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1826), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2074).

```
6.702.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1510), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1575), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

## 6.702 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference 3269

(p. 1827), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2075).

```
6.702.3.5  virtual int activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1511), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1576), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1827), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2075).

```
6.702.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1511), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1576), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1828), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2076).

```
6.702.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1512), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1577), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1828), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 2076).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ResponseMarshaller.h**

## 6.703 **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3260).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::ResponseMarshaller:

## Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.703.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3260).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.703.2 Constructor & Destructor Documentation

6.703.2.1 **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::ResponseMarshaller**  
( ) [inline]

6.703.2.2 `virtual activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::~~ResponseMarshaller ( ) [inline, virtual]`

### 6.703.3 Member Function Documentation

6.703.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1517), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1558), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1806), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2058).

6.703.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

Reimplemented in `activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller` (p. 1518), `activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller` (p. 1558), `activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller` (p. 1806), and `activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller` (p. 2058).

6.703.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

## 6.703 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller Class Reference 3273

<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1518), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1806), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2058).

```
6.703.3.4 virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1518), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1807), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2059).

```
6.703.3.5  virtual int activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1519), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1807), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2059).

```
6.703.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--



Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1519), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1560), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1808), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2060).

```
6.703.3.7 virtual void activemq::wireformat::openwire::marshal::v6::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller** (p. 1520), **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller** (p. 1560), **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller** (p. 1808), and **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller** (p. 2060).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ResponseMarshaller.h**

## 6.704 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

```
#include <src/main/decaf/lang/Runnable.h>
```

Inheritance diagram for decaf::lang::Runnable:

## Public Member Functions

- virtual `~Runnable()`
- virtual void `run()`=0

*Run method - called by the **Thread** (p. 3707) class in the context of the thread.*

### 6.704.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

### 6.704.2 Constructor & Destructor Documentation

6.704.2.1 virtual `decaf::lang::Runnable::~~Runnable()` [`inline`, `virtual`]

### 6.704.3 Member Function Documentation

6.704.3.1 virtual void `decaf::lang::Runnable::run()` [`pure virtual`]

Run method - called by the **Thread** (p. 3707) class in the context of the thread.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1196), **activemq::threads::DedicatedTaskRunner** (p. 1639), **activemq::transport::inactivity::ReadChecker** (p. 3108), **activemq::transport::inactivity::WriteChecker** (p. 3951), **activemq::transport::IOTransport** (p. 2111), **activemq::transport::mock::InternalCommandListener** (p. 2086), **decaf::lang::Thread** (p. 3713), and **decaf::util::concurrent::PooledThread** (p. 2919).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Runnable.h`

## 6.705 decaf::lang::Runtime Class Reference

```
#include <src/main/decaf/lang/Runtime.h>
```

Inheritance diagram for `decaf::lang::Runtime`:

## Public Member Functions

- virtual `~Runtime()`

## Static Public Member Functions

- static `Runtime * getRuntime()`

*Gets the single instance of the Decaf **Runtime** (p. 3265) for this Process.*

- static void **initializeRuntime** (int argc, char \*\*argv)

*Initialize the Decaf Library passing it the args that were passed to the application at startup.*

- static void **initializeRuntime** ()

*Initialize the Decaf Library.*

- static void **shutdownRuntime** ()

*Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.*

## 6.705.1 Constructor & Destructor Documentation

6.705.1.1 virtual decaf::lang::Runtime::~~Runtime ( ) [inline, virtual]

## 6.705.2 Member Function Documentation

6.705.2.1 static Runtime\* decaf::lang::Runtime::getRuntime ( ) [static]

Gets the single instance of the Decaf **Runtime** (p. 3265) for this Process.

### Returns

pointer to the single Decaf **Runtime** (p. 3265) instance that exists for this process

6.705.2.2 static void decaf::lang::Runtime::initializeRuntime ( int argc, char \*\* argv )  
[static]

Initialize the Decaf Library passing it the args that were passed to the application at startup.

### Parameters

<i>argc</i>	- The number of args passed
<i>argv</i>	- Array of char* values passed to the Process on start.

### Exceptions

<i>runtime_error</i>	if the library is already initialized or an error occurs during initialization.
----------------------	---

6.705.2.3 static void decaf::lang::Runtime::initializeRuntime ( ) [static]

Initialize the Decaf Library.

**Exceptions**

<i>runtime_error</i>	if the library is already initialized or an error occurs during initialization.
----------------------	---

**6.705.2.4 static void decaf::lang::Runtime::shutdownRuntime ( ) [static]**

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

**Exceptions**

<i>runtime_error</i>	if the library has not already been initialized or an error occurs during shutdown.
----------------------	---

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runtime.h**

**6.706 decaf::lang::exceptions::RuntimeException Class Reference**

```
#include <src/main/decaf/lang/exceptions/RuntimeException.h>
```

Inheritance diagram for decaf::lang::exceptions::RuntimeException:

**Public Member Functions**

- **RuntimeException** ( ) throw ( )  
*Default Constructor.*
- **RuntimeException** (const **Exception** &ex) throw ( )  
*Conversion Constructor from some other ActiveMQException.*
- **RuntimeException** (const **RuntimeException** &ex) throw ( )  
*Copy Constructor.*
- **RuntimeException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ( )  
*Constructor - Initializes the file name and line number where this message occurred.*
- **RuntimeException** (const std::exception \***cause**) throw ( )  
*Constructor.*
- **RuntimeException** (const char \*file, const int lineNumber, const char \*msg,...) throw ( )  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **RuntimeException** \* **clone** ( ) const  
*Clones this exception.*
- virtual ~**RuntimeException** ( ) throw ( )

## 6.706.1 Constructor & Destructor Documentation

6.706.1.1 `decaf::lang::exceptions::RuntimeException::RuntimeException ( ) throw ()`  
`[inline]`

Default Constructor.

6.706.1.2 `decaf::lang::exceptions::RuntimeException::RuntimeException ( const Exception & ex ) throw ()` `[inline]`

Conversion Constructor from some other ActiveMQException.

### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.706.1.3 `decaf::lang::exceptions::RuntimeException::RuntimeException ( const RuntimeException & ex ) throw ()` `[inline]`

Copy Constructor.

### Parameters

<i>ex</i>	The <b>Exception</b> (p. 1794) whose data is to be copied into this one.
-----------	--

6.706.1.4 `decaf::lang::exceptions::RuntimeException::RuntimeException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.706.1.5 `decaf::lang::exceptions::RuntimeException::RuntimeException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

**Parameters**

<i>cause</i>	<b>Pointer</b> (p. 2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	---

6.706.1.6 `decaf::lang::exceptions::RuntimeException::RuntimeException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.706.1.7 `virtual decaf::lang::exceptions::RuntimeException::~~RuntimeException ( ) throw ()`  
`[inline, virtual]`

**6.706.2 Member Function Documentation**

6.706.2.1 `virtual RuntimeException* decaf::lang::exceptions::RuntimeException::clone ( )`  
`const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

an new **Exception** (p. 1794) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/RuntimeException.h`

**6.707 decaf::security::SecureRandom Class Reference**

```
#include <src/main/decaf/security/SecureRandom.h>
```

Inheritance diagram for decaf::security::SecureRandom:

## Public Member Functions

- **SecureRandom** ()

*Creates a new instance of a secure random number generator that implements the default random number algorithm.*

- **SecureRandom** (const std::vector< unsigned char > &seed)

*Creates a new instance of a secure random number generator that implements the default random number algorithm.*

- **SecureRandom** (const unsigned char \*seed, int size)

*Creates a new instance of a secure random number generator that implements the default random number algorithm.*

- virtual ~**SecureRandom** ()

- virtual void **nextBytes** (std::vector< unsigned char > &buf)

*Modifies the byte array by a random sequence of bytes generated by this random number generator.*

**Parameters**

buf	non-null array to contain the new random bytes
-----	--

**See also**

**next** (p. 3102)

- virtual void **nextBytes** (unsigned char \*buf, int size)

*Modifies the byte array by a random sequence of bytes generated by this random number generator.*

**Parameters**

buf	non-null array to contain the new random bytes
-----	--

**See also**

**next** (p. 3102)

**Exceptions**

NullPointerException	if buff is NULL
IllegalArgumentException	if size is negative

- virtual void **setSeed** (unsigned long long seed)

*Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.*

**Parameters**

seed	the seed that alters the state of the random number generator
------	---

**See also**

**next** (p. 3102)

**Random()** (p. 3102)

**#Random(long)**

- virtual void **setSeed** (const std::vector< unsigned char > &seed)  
*Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.*
- virtual void **setSeed** (const unsigned char \*seed, int size)  
*Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.*

## Protected Member Functions

- virtual int **next** (int bits)  
*Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E. Knuth in The Art of Computer Programming, Volume 2: Seminumerical Algorithms, section 3.2.1.*

### Returns

*int a pseudo-random generated int number*

### Parameters

bits	number of bits of the returned value
------	--------------------------------------

### See also

**nextBytes** (p. 3103)  
**nextDouble** (p. 3104)  
**nextFloat** (p. 3104)  
**nextInt()** (p. 3104)  
**nextInt(int)** (p. 3105)  
**nextGaussian** (p. 3104)  
**nextLong** (p. 3105)

## 6.707.1 Detailed Description

### Since

1.0

## 6.707.2 Constructor & Destructor Documentation

### 6.707.2.1 decaf::security::SecureRandom::SecureRandom ( )

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3269) instance that is created with this constructor is unseeded and can be seeded by calling the setSeed method. Calls to nextBytes on an unseeded **SecureRandom** (p. 3269) result in the object seeding itself.



6.707.2.2 `decaf::security::SecureRandom::SecureRandom ( const std::vector< unsigned char > & seed )`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3269) instance created by this constructor is seeded using the passed byte array.

#### Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
-------------	--

6.707.2.3 `decaf::security::SecureRandom::SecureRandom ( const unsigned char * seed, int size )`

Creates a new instance of a secure random number generator that implements the default random number algorithm.

The **SecureRandom** (p. 3269) instance created by this constructor is seeded using the passed byte array.

#### Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
<i>size</i>	The number of bytes in the seed buffer.

#### Exceptions

<i>NullPointerException</i>	if the seed buffer is NULL.
<i>IllegalArgumentException</i>	if the size value is negative.

6.707.2.4 `virtual decaf::security::SecureRandom::~~SecureRandom ( ) [virtual]`

### 6.707.3 Member Function Documentation

6.707.3.1 `virtual int decaf::security::SecureRandom::next ( int bits ) [protected, virtual]`

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

#### Returns

`int` a pseudo-random generated `int` number

**Parameters**

<i>bits</i>	number of bits of the returned value
-------------	--------------------------------------

**See also**

**nextBytes** (p. 3103)  
**nextDouble** (p. 3104)  
**nextFloat** (p. 3104)  
**nextInt()** (p. 3104)  
**nextInt(int)** (p. 3105)  
**nextGaussian** (p. 3104)  
**nextLong** (p. 3105)

Reimplemented from **decaf::util::Random** (p. 3102).

6.707.3.2 `virtual void decaf::security::SecureRandom::nextBytes ( unsigned char * buf, int size ) [virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

**Parameters**

<i>buf</i>	non-null array to contain the new random bytes
------------	--

**See also**

**next** (p. 3102)

**Exceptions**

<i>NullPointerException</i>	if buff is NULL
<i>IllegalArgumentException</i>	if size is negative

Reimplemented from **decaf::util::Random** (p. 3103).

6.707.3.3 `virtual void decaf::security::SecureRandom::nextBytes ( std::vector< unsigned char > & buf ) [virtual]`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

**Parameters**

<i>buf</i>	non-null array to contain the new random bytes
------------	--

**See also**

**next** (p. 3102)

Reimplemented from **decaf::util::Random** (p. 3103).

6.707.3.4 `virtual void decaf::security::SecureRandom::setSeed ( const std::vector< unsigned char > & seed ) [virtual]`

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

#### Parameters

<i>seed</i>	A vector of bytes that is used update the seed of the RNG.
-------------	--

6.707.3.5 `virtual void decaf::security::SecureRandom::setSeed ( const unsigned char * seed, int size ) [virtual]`

Supplements or sets the seed of this secure random number generator, calls to this method never reduces randomness.

#### Parameters

<i>seed</i>	The seed bytes to use to seed this secure random number generator.
<i>size</i>	The number of bytes in the seed buffer.

#### Exceptions

<i>NullPointerException</i>	if the seed buffer is NULL.
<i>IllegalArgumentException</i>	if the size value is negative.

6.707.3.6 `virtual void decaf::security::SecureRandom::setSeed ( unsigned long long seed ) [virtual]`

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

#### Parameters

<i>seed</i>	the seed that alters the state of the random number generator
-------------	---

#### See also

**next** (p. 3102)  
**Random()** (p. 3102)  
**#Random(long)**

Reimplemented from **decaf::util::Random** (p. 3105).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SecureRandom.h**

## 6.708 decaf::internal::security::SecureRandomImpl Class Reference

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

```
#include <src/main/decaf/internal/security/unix/SecureRandomImpl.h>
```

Inheritance diagram for decaf::internal::security::SecureRandomImpl:

### Public Member Functions

- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*
- **SecureRandomImpl** ()
- virtual **~SecureRandomImpl** ()
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*

### 6.708.1 Detailed Description

Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.

Secure Random Number Generator for Windows based platforms that attempts to obtain secure bytes with high entropy from known sources.

If the platform does not have a source of secure bytes then the platform random number generator is used if one exists otherwise the Decaf RNG is used as a last resort.

**Since**

1.0

**6.708.2 Constructor & Destructor Documentation****6.708.2.1** decaf::internal::security::SecureRandomImpl::SecureRandomImpl ( )**6.708.2.2** virtual decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ( )  
[virtual]**6.708.2.3** decaf::internal::security::SecureRandomImpl::SecureRandomImpl ( )**6.708.2.4** virtual decaf::internal::security::SecureRandomImpl::~~SecureRandomImpl ( )  
[virtual]**6.708.3 Member Function Documentation****6.708.3.1** virtual unsigned char\* decaf::internal::security::SecureRandomImpl::providerGenerateSeed  
( int numBytes ) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

**Parameters**

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implements **decaf::security::SecureRandomSpi** (p. 3279).

**6.708.3.2** virtual unsigned char\* decaf::internal::security::SecureRandomImpl::providerGenerateSeed  
( int numBytes ) [virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

**Parameters**

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implements **decaf::security::SecureRandomSpi** (p. 3279).

6.708.3.3 `virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes ( unsigned char * bytes, int numBytes ) [virtual]`

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

#### Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 3279).

6.708.3.4 `virtual void decaf::internal::security::SecureRandomImpl::providerNextBytes ( unsigned char * bytes, int numBytes ) [virtual]`

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

#### Parameters

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implements **decaf::security::SecureRandomSpi** (p. 3279).

6.708.3.5 `virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed ( const unsigned char * seed, int size ) [virtual]`

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

#### Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 3279).

6.708.3.6 virtual void decaf::internal::security::SecureRandomImpl::providerSetSeed ( const unsigned char \* *seed*, int *size* ) [virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.

#### Parameters

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implements **decaf::security::SecureRandomSpi** (p. 3279).

The documentation for this class was generated from the following files:

- src/main/decaf/internal/security/unix/**SecureRandomImpl.h**
- src/main/decaf/internal/security/windows/**SecureRandomImpl.h**

## 6.709 decaf::security::SecureRandomSpi Class Reference

Interface class used by Security Service Providers to implement a source of secure random bytes.

```
#include <src/main/decaf/security/SecureRandomSpi.h>
```

Inheritance diagram for decaf::security::SecureRandomSpi:

### Public Member Functions

- **SecureRandomSpi** ()
- virtual ~**SecureRandomSpi** ()
- virtual void **providerSetSeed** (const unsigned char \*seed, int size)=0  
*Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.*
- virtual void **providerNextBytes** (unsigned char \*bytes, int numBytes)=0  
*Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.*
- virtual unsigned char \* **providerGenerateSeed** (int numBytes)=0  
*Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.*

### 6.709.1 Detailed Description

Interface class used by Security Service Providers to implement a source of secure random bytes.

**Since**

1.0

**6.709.2 Constructor & Destructor Documentation**6.709.2.1 `decaf::security::SecureRandomSpi::SecureRandomSpi ( )`6.709.2.2 `virtual decaf::security::SecureRandomSpi::~~SecureRandomSpi ( )` [virtual]**6.709.3 Member Function Documentation**6.709.3.1 `virtual unsigned char* decaf::security::SecureRandomSpi::providerGenerateSeed ( int numBytes )` [pure virtual]

Generates a new set of seed bytes, the returned value may be used to seed another Random Number Generator.

The caller owns the returned array and must delete it.

**Parameters**

<i>numBytes</i>	The number of bytes that should be generated for the new seed array.
-----------------	--

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3276), and `decaf::internal::security::SecureRandomSpi` (p. 3276).

6.709.3.2 `virtual void decaf::security::SecureRandomSpi::providerNextBytes ( unsigned char * bytes, int numBytes )` [pure virtual]

Generates the number of random bytes specified by the size parameter and write them to the passed bytes array.

The array must have already been allocated and be of the correct size to prevent segmentation faults.

**Parameters**

<i>bytes</i>	The array that will be filled with random bytes equal to size.
<i>numBytes</i>	The number of bytes to generate and write into the bytes array.

Implemented in `decaf::internal::security::SecureRandomImpl` (p. 3277), and `decaf::internal::security::SecureRandomSpi` (p. 3277).

6.709.3.3 `virtual void decaf::security::SecureRandomSpi::providerSetSeed ( const unsigned char * seed, int size )` [pure virtual]

Reseed the Random Number Generator, the value given supplements the previous seed value if any instead of replacing it.



**Parameters**

<i>seed</i>	The array of bytes used to update the generators seed.
<i>size</i>	The size of the passed byte array.

Implemented in **decaf::internal::security::SecureRandomImpl** (p. 3278), and **decaf::internal::security::SecureRandomImpl** (p. 3278).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SecureRandomSpi.h**

**6.710 decaf::util::concurrent::Semaphore Class Reference**

A counting semaphore.

```
#include <src/main/decaf/util/concurrent/Semaphore.h>
```

**Public Member Functions**

- **Semaphore** (int permits)  
*Creates a **Semaphore** (p. 3280) with the given number of permits and nonfair fairness setting.*
- **Semaphore** (int permits, bool fair)  
*Creates a **Semaphore** (p. 3280) with the given number of permits and the given fairness setting.*
- virtual **~Semaphore** ()
- void **acquire** () throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.*
- void **acquireUninterruptibly** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, blocking until one is available.*
- bool **tryAcquire** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, only if one is available at the time of invocation.*
- bool **tryAcquire** (long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )  
*Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.*
- void **release** () throw ( decaf::lang::exceptions::RuntimeException )  
*Releases a permit, returning it to the semaphore.*
- void **acquire** (int permits) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.*

- void **acquireUninterruptibly** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, blocking until all are available.*
- bool **tryAcquire** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.*
- bool **tryAcquire** (int permits, long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.*
- void **release** (int permits) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )  
*Releases the given number of permits, returning them to the semaphore.*
- int **availablePermits** () const  
*Returns the current number of permits available in this semaphore.*
- int **drainPermits** () throw ( decaf::lang::exceptions::RuntimeException )  
*Acquires and returns all permits that are immediately available.*
- bool **isFair** () const
- std::string **toString** () const  
*Returns a string identifying this semaphore, as well as its state.*

### 6.710.1 Detailed Description

A counting semaphore.

Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 3283) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 3286) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 3280) just keeps a count of the number available and acts accordingly.

Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

```
class Pool { private:
```

```
static const int MAX_AVAILABLE = 100; Semaphore (p. 3280) available;
```

```
std::vector<std::string> items; std::vector<bool> used;
```

```
Mutex (p. 2736) lock;
```

```
public:
```

```
Pool() : available( MAX_AVAILABLE, true ) { used.resize( MAX_AVAILABLE ); items.resize( MAX_AVAILABLE ); }
```

```
std::string getItem() throws InterruptedException { available.acquire(); return getNextAvailableItem(); }
```

```
void putItem( std::string x ) { if( markAsUnused(x) ) { available.release(); } }

std::string getNextAvailableItem() {

synchronized( &lock ) (p. 4511) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( !used[i]
) { used[i] = true; return items[i]; } }

return std::string(); // not reached }

bool markAsUnused( const std::string& item ) { synchronized( &lock ) (p. 4511) { for(
int i = 0; i < MAX_AVAILABLE; ++i ) { if( item == items[i] ) { if( used[i] ) { used[i] = false;
return true; } else return false; } } } return false; } };
```

Before obtaining an item each thread must acquire a permit from the semaphore, guaranteeing that an item is available for use. When the thread has finished with the item it is returned back to the pool and a permit is returned to the semaphore, allowing another thread to acquire that item. Note that no synchronization lock is held when **acquire()** (p. 3283) is called as that would prevent an item from being returned to the pool. The semaphore encapsulates the synchronization needed to restrict access to the pool, separately from any synchronization needed to maintain the consistency of the pool itself.

A semaphore initialized to one, and which is used such that it only has at most one permit available, can serve as a mutual exclusion lock. This is more commonly known as a binary semaphore, because it only has two states: one permit available, or zero permits available. When used in this way, the binary semaphore has the property (unlike many **Lock** (p. 2334) implementations), that the "lock" can be released by a thread other than the owner (as semaphores have no notion of ownership). This can be useful in some specialized contexts, such as deadlock recovery.

The constructor for this class optionally accepts a fairness parameter. When set false, this class makes no guarantees about the order in which threads acquire permits. In particular, barging is permitted, that is, a thread invoking **acquire()** (p. 3283) can be allocated a permit ahead of a thread that has been waiting - logically the new thread places itself at the head of the queue of waiting threads. When fairness is set true, the semaphore guarantees that threads invoking any of the acquire methods are selected to obtain permits in the order in which their invocation of those methods was processed (first-in-first-out; FIFO). Note that FIFO ordering necessarily applies to specific internal points of execution within these methods. So, it is possible for one thread to invoke acquire before another, but reach the ordering point after the other, and similarly upon return from the method. Also note that the untimed tryAcquire methods do not honor the fairness setting, but will take any permits that are available.

Generally, semaphores used to control resource access should be initialized as fair, to ensure that no thread is starved out from accessing a resource. When using semaphores for other kinds of synchronization control, the throughput advantages of non-fair ordering often outweigh fairness considerations.

This class also provides convenience methods to acquire and release multiple permits at a time. Beware of the increased risk of indefinite postponement when these methods are used without fairness set true.

#### Since

1.0

## 6.710.2 Constructor & Destructor Documentation

### 6.710.2.1 `decaf::util::concurrent::Semaphore::Semaphore ( int permits )`

Creates a **Semaphore** (p. 3280) with the given number of permits and nonfair fairness setting.

#### Parameters

<i>permits</i>	the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.
----------------	---

### 6.710.2.2 `decaf::util::concurrent::Semaphore::Semaphore ( int permits, bool fair )`

Creates a **Semaphore** (p. 3280) with the given number of permits and the given fairness setting.

#### Parameters

<i>permits</i>	the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.
<i>fair</i>	true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

### 6.710.2.3 `virtual decaf::util::concurrent::Semaphore::~~Semaphore ( )` [virtual]

## 6.710.3 Member Function Documentation

### 6.710.3.1 `void decaf::util::concurrent::Semaphore::acquire ( ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* Some other thread invokes the **release()** (p. 3286) method for this semaphore and the current thread is next to be assigned a permit; or \* Some other thread interrupts the current thread.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

### Exceptions

<i>InterruptedException</i>	- if the current thread is interrupted.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).

```
6.710.3.2 void decaf::util::concurrent::Semaphore::acquire ( int permits )  
          throw ( decaf::lang::exceptions::InterruptedException,  
                decaf::lang::exceptions::IllegalArgumentException,  
                decaf::lang::exceptions::RuntimeException )
```

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

\* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or \* Some other thread interrupts the current thread.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting for a permit,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to **release()** (p. 3286).

### Parameters

<i>permits</i>	the number of permits to acquire.
----------------	-----------------------------------

### Exceptions

<i>InterruptedException</i>	if the current thread is interrupted.
<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).

6.710.3.3 `void decaf::util::concurrent::Semaphore::acquireUninterruptibly ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, blocking until one is available.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes the **release()** (p. 3286) method for this semaphore and the current thread is next to be assigned a permit.

If the current thread is interrupted while waiting for a permit then it will continue to wait, but the time at which the thread is assigned a permit may change compared to the time it would have received the permit had no interruption occurred. When the thread does return from this method its interrupt status will be set.

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).
-------------------------	---

6.710.3.4 `void decaf::util::concurrent::Semaphore::acquireUninterruptibly ( int permits ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, blocking until all are available.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request.

If the current thread is interrupted while waiting for permits then it will continue to wait and its position in the queue is not affected. When the thread does return from this method its interrupt status will be set.

#### Parameters

<i>permits</i>	the number of permits to acquire.
----------------	-----------------------------------

#### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).

6.710.3.5 `int decaf::util::concurrent::Semaphore::availablePermits ( ) const`

Returns the current number of permits available in this semaphore.

This method is typically used for debugging and testing purposes.

#### Returns

the number of permits available in this semaphore

6.710.3.6 `int decaf::util::concurrent::Semaphore::drainPermits ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Acquires and returns all permits that are immediately available.

#### Returns

the number of permits acquired

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while draining the <b>Semaphore</b> (p. 3280).
-------------------------	--

6.710.3.7 `bool decaf::util::concurrent::Semaphore::isFair ( ) const`

#### Returns

true if this semaphore has fairness set true

6.710.3.8 `void decaf::util::concurrent::Semaphore::release ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Releases a permit, returning it to the semaphore.

Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling **acquire()** (p. 3283). Correct usage of a semaphore is established by programming convention in the application.

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while releasing the <b>Semaphore</b> (p. 3280).
-------------------------	---

```
6.710.3.9 void decaf::util::concurrent::Semaphore::release ( int permits )
          throw ( decaf::lang::exceptions::IllegalArgumentException,
                  decaf::lang::exceptions::RuntimeException )
```

Releases the given number of permits, returning them to the semaphore.

Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes; otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

#### Parameters

<i>permits</i>	the number of permits to release
----------------	----------------------------------

#### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while releasing the <b>Semaphore</b> (p. 3280).

```
6.710.3.10 std::string decaf::util::concurrent::Semaphore::toString ( ) const
```

Returns a string identifying this semaphore, as well as its state.

The state, in brackets, includes the String "Permits =" followed by the number of permits.

#### Returns

a string identifying this semaphore, as well as its state

```
6.710.3.11 bool decaf::util::concurrent::Semaphore::tryAcquire ( int
                    permits, long long timeout, const TimeUnit & unit ) throw
          ( decaf::lang::exceptions::IllegalArgumentException,
            decaf::lang::exceptions::RuntimeException )
```

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

Acquires the given number of permits, if they are available and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this



request; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If the permits are acquired then the value true is returned.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting to acquire the permits,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3286).

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 3286).

#### Parameters

<i>permits</i>	the number of permits to acquire
<i>timeout</i>	the maximum amount of time to wait to acquire the permits.
<i>unit</i>	the units that the timeout param represents.

#### Returns

true if all permits were acquired and false if the waiting time elapsed before all permits were acquired

#### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).

6.710.3.12 **bool** decaf::util::concurrent::Semaphore::tryAcquire ( long long *timeout*, const TimeUnit & *unit* ) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException )

Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

\* Some other thread invokes the **release()** (p. 3286) method for this semaphore and the current thread is next to be assigned a permit; or \* Some other thread interrupts the current thread; or \* The specified waiting time elapses.

If a permit is acquired then the value `true` is returned.

If the current thread:

\* has its interrupted status set on entry to this method; or \* is interrupted while waiting to acquire a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value `false` is returned. If the time is less than or equal to zero, the method will not wait at all.

#### Parameters

<i>timeout</i>	the maximum time to wait for a permit
<i>unit</i>	the time unit of the timeout argument

#### Returns

`true` if a permit was acquired and `false` if the waiting time elapsed before a permit was acquired

#### Exceptions

<i>InterruptedException</i>	if the current thread is interrupted.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).

6.710.3.13 `bool decaf::util::concurrent::Semaphore::tryAcquire ( ) throw ( decaf::lang::exceptions::RuntimeException )`

Acquires a permit from this semaphore, only if one is available at the time of invocation.

Acquires a permit, if one is available and returns immediately, with the value `true`, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value `false`.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire()** (p. 3289) will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use `tryAcquire(0, TimeUnit.SECONDS` (p. 3757)) which is almost equivalent (it also detects interruption).

#### Returns

`true` if a permit was acquired and `false` otherwise

#### Exceptions

<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).
-------------------------	---

6.710.3.14 `bool decaf::util::concurrent::Semaphore::tryAcquire ( int permits )  
 throw ( decaf::lang::exceptions::IllegalArgumentException,  
 decaf::lang::exceptions::RuntimeException )`

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.

Acquires the given number of permits, if they are available, and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then this method will return immediately with the value false and the number of available permits is unchanged.

Even when this semaphore has been set to use a fair ordering policy, a call to tryAcquire will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use tryAcquire(permits, 0, **TimeUnit.SECONDS** (p. 3757)) which is almost equivalent (it also detects interruption).

#### Parameters

<i>permits</i>	the number of permits to acquire
----------------	----------------------------------

#### Returns

true if the permits were acquired and false otherwise.

#### Exceptions

<i>IllegalArgumentException</i>	if the permits argument is negative.
<i>RuntimeException</i>	if an unexpected error occurs while acquiring the <b>Semaphore</b> (p. 3280).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Semaphore.h**

## 6.711 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::SendExecutor:

## Public Member Functions

- **SendExecutor** (**MessageCreator** \*messageCreator, **CmsTemplate** \*parent)
- virtual **~SendExecutor** ()
- virtual void **dolnCms** (**cms::Session** \*session, **cms::MessageProducer** \*producer) throw ( cms::CMSException )

*Execute an action given a session and producer.*

## Protected Member Functions

- **SendExecutor** (const **SendExecutor** &)
- **SendExecutor & operator=** (const **SendExecutor** &)

### 6.711.1 Constructor & Destructor Documentation

6.711.1.1 **activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor** ( const **SendExecutor** & ) [inline, protected]

6.711.1.2 **activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor** ( **MessageCreator** \* messageCreator, **CmsTemplate** \* parent ) [inline]

6.711.1.3 virtual **activemq::cmsutil::CmsTemplate::SendExecutor::~~SendExecutor** ( ) [inline, virtual]

### 6.711.2 Member Function Documentation

6.711.2.1 virtual void **activemq::cmsutil::CmsTemplate::SendExecutor::dolnCms** ( **cms::Session** \* session, **cms::MessageProducer** \* producer ) throw ( **cms::CMSException** ) [inline, virtual]

Execute an action given a session and producer.

#### Parameters

<i>session</i>	the CMS Session
<i>producer</i>	the CMS Producer

#### Exceptions

<b>cms::CMSException</b> (p. 1130)	if thrown by CMS API methods
---------------------------------------	------------------------------

Implements **activemq::cmsutil::ProducerCallback** (p. 3012).

### 6.711.2.2 SendExecutor& activemq::cmsutil::CmsTemplate::SendExecutor::operator= ( const SendExecutor & ) [inline, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

## 6.712 decaf::net::ServerSocket Class Reference

This class implements server sockets.

```
#include <src/main/decaf/net/ServerSocket.h>
```

Inheritance diagram for decaf::net::ServerSocket:

### Public Member Functions

- **ServerSocket** ()  
*Creates a non-bound server socket.*
- **ServerSocket** (int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **ServerSocket** (int port, int backlog, const InetAddress \*address) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- virtual ~**ServerSocket** ()  
*Releases socket handle if **close()** (p. 3297) hasn't been called.*
- virtual void **bind** (const std::string &host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.*
- virtual void **bind** (const std::string &host, int port, int backlog) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.*
- virtual **Socket** \* **accept** () throw ( decaf::io::IOException )

*Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3292), the caller blocks until a connection is made.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.*

- virtual bool **isClosed** () const
- virtual bool **isBound** () const
- virtual int **getReceiveBufferSize** () const throw ( SocketException )

*Gets the receive buffer size for this socket, SO\_RCVBUF.*

- virtual void **setReceiveBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )

*Sets the receive buffer size for this socket, SO\_RCVBUF.*

- virtual bool **getReuseAddress** () const throw ( SocketException )

*Gets the reuse address flag, SO\_REUSEADDR.*

- virtual void **setReuseAddress** (bool reuse) throw ( SocketException )

*Sets the reuse address flag, SO\_REUSEADDR.*

- virtual int **getSoTimeout** () const throw ( SocketException )

*Gets the timeout for socket operations, SO\_TIMEOUT.*

- virtual void **setSoTimeout** (int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )

*Sets the timeout for socket operations, SO\_TIMEOUT.*

- virtual int **getLocalPort** () const

*Gets the port number on the Local machine that this **ServerSocket** (p. 3292) is bound to.*

- virtual std::string **toString** () const

## Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory \*factory) throw ( decaf::io::IOException, decaf::net::SocketException )

*Sets the instance of a **SocketImplFactory** (p. 3481) that the **ServerSocket** (p. 3292) class should use when new instances of this class are created.*

## Protected Member Functions

- **ServerSocket** (SocketImpl \*impl)

*Creates a **ServerSocket** (p. 3292) wrapping the provided **SocketImpl** (p. 3472) instance, this **Socket** (p. 3445) is considered unconnected.*

- virtual void **implAccept** (Socket \*socket) throw ( decaf::io::IOException )

*Virtual method that allows a **ServerSocket** (p. 3292) subclass to override the accept call and provide its own **SocketImpl** (p. 3472) for the socket.*

- virtual int **getDefaultBacklog** ()

*Allows a subclass to override what is considered the default backlog.*

- void **checkClosed** () const throw ( decaf::io::IOException )
- void **ensureCreated** () const throw ( decaf::io::IOException )
- void **setupSocketImpl** (int port, int backlog, const InetAddress \*ifAddress)

### 6.712.1 Detailed Description

This class implements server sockets.

A server socket waits for requests to come in over the network.

The actual work of the server socket is performed by an instance of the **SocketImpl** (p. 3472) class. An application can change the socket factory that creates the socket implementation to configure itself to create sockets of a particular type.

#### Since

1.0

### 6.712.2 Constructor & Destructor Documentation

#### 6.712.2.1 decaf::net::ServerSocket::ServerSocket ( )

Creates a non-bound server socket.

#### 6.712.2.2 decaf::net::ServerSocket::ServerSocket ( int *port* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3481) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3472) is created.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
-------------	--

#### Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentException</i>	if the port value is negative or greater than 65535.

#### 6.712.2.3 decaf::net::ServerSocket::ServerSocket ( int *port*, int *backlog* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )

Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connec-

tions that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3481) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3472) is created.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.

#### Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentException</i>	if the port value is negative or greater than 65535.

**6.712.2.4** `decaf::net::ServerSocket::ServerSocket ( int port, int backlog, const InetAddress * address ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )`

Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the `ifAddress` is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at `backlog`, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3481) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3472) is created.

#### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.
<i>ifAddress</i>	The IP Address to bind to on the local machine.

#### Exceptions

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentException</i>	if the port value is negative or greater than 65535.

**6.712.2.5** `virtual decaf::net::ServerSocket::~ServerSocket ( ) [virtual]`

Releases socket handle if `close()` (p. 3297) hasn't been called.



## 6.712.2.6 decaf::net::ServerSocket::ServerSocket ( SocketImpl \* impl ) [protected]

Creates a **ServerSocket** (p. 3292) wrapping the provided **SocketImpl** (p. 3472) instance, this **Socket** (p. 3445) is considered unconnected.

The **ServerSocket** (p. 3292) class takes ownership of this **SocketImpl** (p. 3472) pointer and will delete it when the **Socket** (p. 3445) class is destroyed.

## Parameters

<i>impl</i>	The <b>SocketImpl</b> (p. 3472) instance to wrap.
-------------	---

## Exceptions

<i>NullPointerException</i>	if the passed <b>SocketImpl</b> (p. 3472) is Null.
-----------------------------	--

## 6.712.3 Member Function Documentation

## 6.712.3.1 virtual Socket\* decaf::net::ServerSocket::accept ( ) throw ( decaf::io::IOException ) [virtual]

Listens for a connection request on the bound IPAddress and Port for this **ServerSocket** (p. 3292), the caller blocks until a connection is made.

If the SO\_TIMEOUT option is set this method could throw a **SocketTimeoutException** (p. 3487) if the operation times out.

## Returns

a new **Socket** (p. 3445) object pointer. Never returns NULL, the returned pointer is owned by the caller and must be explicitly freed by them.

## Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
<b>SocketException</b> (p. 3465)	if an error occurs while blocking on the accept call.
<b>SocketTimeoutException</b> (p. 3487)	if the SO_TIMEOUT option was used and the accept timed out.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2799).

## 6.712.3.2 virtual void decaf::net::ServerSocket::bind ( const std::string &amp; host, int port, int backlog ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Bind and listen to given local IPAddress and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

If the backlog is greater than zero it will be used instead of the default value, otherwise the default value is used and no error is generated.

#### Parameters

<i>host</i>	The IP address or host name.
<i>port</i>	The TCP port between 1..65535.
<i>backlog</i>	The size of listen backlog.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
<i>IllegalArgumentException</i>	if the parameters are not valid.

```
6.712.3.3  virtual void decaf::net::ServerSocket::bind ( const std::string
              & host, int port ) throw ( decaf::io::IOException,
              decaf::lang::exceptions::IllegalArgumentException ) [virtual]
```

Bind and listen to given local IP address and port, if the address is empty than a valid local address will be chosen, and if the port of 0 than an available open port will be chosen.

#### Parameters

<i>host</i>	The IP address or host name.
<i>port</i>	The TCP port between 1..65535.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while binding the socket.
<i>IllegalArgumentException</i>	if the parameters are not valid.

```
6.712.3.4  void decaf::net::ServerSocket::checkClosed ( ) const throw (
              decaf::io::IOException ) [protected]
```

```
6.712.3.5  virtual void decaf::net::ServerSocket::close ( ) throw ( decaf::io::IOException )
              [virtual]
```

Closes the server socket, causing any Threads blocked on an accept call to throw an Exception.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

6.712.3.6 void decaf::net::ServerSocket::ensureCreated ( ) const throw ( decaf::io::IOException ) [protected]

6.712.3.7 virtual int decaf::net::ServerSocket::getDefaultBacklog ( ) [protected, virtual]

Allows a subclass to override what is considered the default backlog.

#### Returns

the default backlog for connections.

6.712.3.8 virtual int decaf::net::ServerSocket::getLocalPort ( ) const [virtual]

Gets the port number on the Local machine that this **ServerSocket** (p. 3292) is bound to.

#### Returns

the port number of this machine that is bound, if not bound returns -1.

6.712.3.9 virtual int decaf::net::ServerSocket::getReceiveBufferSize ( ) const throw ( SocketException ) [virtual]

Gets the receive buffer size for this socket, SO\_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

#### Returns

the receive buffer size in bytes.

#### Exceptions

<b>SocketException</b> (p. 3465)	if the operation fails.
-------------------------------------	-------------------------

6.712.3.10 virtual bool decaf::net::ServerSocket::getReuseAddress ( ) const throw ( SocketException ) [virtual]

Gets the reuse address flag, SO\_REUSEADDR.

#### Returns

True if the address can be reused.

#### Exceptions

<b><i>SocketException</i></b> (p. 3465)	if the operation fails.
--	-------------------------

6.712.3.11 `virtual int decaf::net::ServerSocket::getSoTimeout ( ) const throw ( SocketException ) [virtual]`

Gets the timeout for socket operations, SO\_TIMEOUT.

#### Returns

The timeout in milliseconds for socket operations.

#### Exceptions

<b><i>SocketException</i></b> (p. 3465)	Thrown if unable to retrieve the information.
--	---

6.712.3.12 `virtual void decaf::net::ServerSocket::implAccept ( Socket * socket ) throw ( decaf::io::IOException ) [protected, virtual]`

Virtual method that allows a **ServerSocket** (p. 3292) subclass to override the accept call and provide its own **SocketImpl** (p. 3472) for the socket.

#### Parameters

<i>socket</i>	The socket object whose <b>SocketImpl</b> (p. 3472) should be used for the accept call.
---------------	---

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

6.712.3.13 `virtual bool decaf::net::ServerSocket::isBound ( ) const [virtual]`

#### Returns

true if the server socket is bound.

6.712.3.14 `virtual bool decaf::net::ServerSocket::isClosed ( ) const [virtual]`

#### Returns

true if the close method has been called on the **ServerSocket** (p. 3292).

6.712.3.15 virtual void decaf::net::ServerSocket::setReceiveBufferSize ( int *size* ) throw ( **SocketException**, decaf::lang::exceptions::IllegalArgumentException )  
[virtual]

Sets the receive buffer size for this socket, SO\_RCVBUF.

#### Parameters

<i>size</i>	Number of bytes to set the receive buffer to.
-------------	---

#### Exceptions

<b>SocketException</b> (p. 3465)	if the operation fails.
<b>IllegalArgumentException</b>	if the value is zero or negative.

6.712.3.16 virtual void decaf::net::ServerSocket::setReuseAddress ( bool *reuse* ) throw ( **SocketException** ) [virtual]

Sets the reuse address flag, SO\_REUSEADDR.

#### Parameters

<i>reuse</i>	If true, sets the flag.
--------------	-------------------------

#### Exceptions

<b>SocketException</b> (p. 3465)	if the operation fails.
-------------------------------------	-------------------------

6.712.3.17 static void decaf::net::ServerSocket::setSocketImplFactory ( **SocketImplFactory** \* *factory* ) throw ( decaf::io::IOException, decaf::net::SocketException )  
[static]

Sets the instance of a **SocketImplFactory** (p. 3481) that the **ServerSocket** (p. 3292) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

#### Parameters

<i>factory</i>	The instance of a <b>SocketImplFactory</b> (p. 3481) to use when new <b>SocketImpl</b> (p. 3472) objects are created.
----------------	---

#### Exceptions

<b>IOException</b>	if an I/O error occurs while performing this operation.
<b>SocketException</b> (p. 3465)	if this method has already been called with a valid factory.

6.712.3.18 `virtual void decaf::net::ServerSocket::setSoTimeout ( int timeout ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
`[virtual]`

Sets the timeout for socket operations, SO\_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

#### Parameters

<i>timeout</i>	The timeout in milliseconds for socket operations.
----------------	--

#### Exceptions

<b>SocketException</b> (p. 3465)	Thrown if unable to set the information.
<i>IllegalArgumentException</i>	if the timeout value is negative.

6.712.3.19 `void decaf::net::ServerSocket::setupSocketImpl ( int port, int backlog, const InetAddress * ifAddress )` `[protected]`

6.712.3.20 `virtual std::string decaf::net::ServerSocket::toString ( ) const` `[virtual]`

#### Returns

a string representing this **ServerSocket** (p. 3292).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocket.h**

## 6.713 decaf::net::ServerSocketFactory Class Reference

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

```
#include <src/main/decaf/net/ServerSocketFactory.h>
```

Inheritance diagram for decaf::net::ServerSocketFactory:

#### Public Member Functions

- virtual **~ServerSocketFactory** ()
- virtual **ServerSocket** \* **createServerSocket** ()  
*Create a new **ServerSocket** (p. 3292) that is unbound.*

- virtual **ServerSocket** \* **createServerSocket** (int port)=0  
Create a new **ServerSocket** (p. 3292) that is bound to the given port.
- virtual **ServerSocket** \* **createServerSocket** (int port, int backlog)=0  
Create a new **ServerSocket** (p. 3292) that is bound to the given port.
- virtual **ServerSocket** \* **createServerSocket** (int port, int backlog, const **InetAddress** \*address)=0  
Create a new **ServerSocket** (p. 3292) that is bound to the given port.

### Static Public Member Functions

- static **ServerSocketFactory** \* **getDefault** ()  
Returns the Default **ServerSocket** (p. 3292) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.

### Protected Member Functions

- **ServerSocketFactory** ()

#### 6.713.1 Detailed Description

Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.

#### Since

1.0

#### 6.713.2 Constructor & Destructor Documentation

6.713.2.1 **decaf::net::ServerSocketFactory::ServerSocketFactory** ( ) [protected]

6.713.2.2 virtual **decaf::net::ServerSocketFactory::~~ServerSocketFactory** ( ) [virtual]

#### 6.713.3 Member Function Documentation

6.713.3.1 virtual **ServerSocket**\* **decaf::net::ServerSocketFactory::createServerSocket** ( )  
[virtual]

Create a new **ServerSocket** (p. 3292) that is unbound.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

#### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Reimplemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1650), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1660), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2805).

6.713.3.2 `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( int port ) [pure virtual]`

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
-------------	--

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

**Exceptions**

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1651), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1660), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2805).

6.713.3.3 `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( int port, int backlog, const InetAddress * address ) [pure virtual]`

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory. The **ServerSocket** (p. 3292) will bind to the specified interface on the local host, and accept connections only on that interface. If the address parameter is NULL than the **ServerSocket** (p. 3292) will listen on all interfaces.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.
<i>address</i>	The address of the interface on the local machine to bind to.

**Returns**

new **ServerSocket** (p. 3292) pointer that is owned by the caller.



### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1650), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1661), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2806).

**6.713.3.4** `virtual ServerSocket* decaf::net::ServerSocketFactory::createServerSocket ( int port, int backlog ) [pure virtual]`

Create a new **ServerSocket** (p. 3292) that is bound to the given port.

The **ServerSocket** (p. 3292) will have been configured with the defaults from the factory.  
The **ServerSocket** (p. 3292) will use the specified connection backlog setting.

### Parameters

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The number of pending connect request the <b>ServerSocket</b> (p. 3292) can queue.

### Returns

new **ServerSocket** (p. 3292) pointer that is owned by the caller.

### Exceptions

<i>IOException</i>	if the <b>ServerSocket</b> (p. 3292) cannot be created for some reason.
--------------------	---

Implemented in **decaf::internal::net::DefaultServerSocketFactory** (p. 1651), **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1661), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2806).

**6.713.3.5** `static ServerSocketFactory* decaf::net::ServerSocketFactory::getDefault ( ) [static]`

Returns the Default **ServerSocket** (p. 3292) factory, the pointer is owned by the Decaf runtime and should not be deleted by the caller.

Only one default **ServerSocketFactory** (p. 3301) exists for the lifetime of the Application.

### Returns

the default **ServerSocketFactory** (p. 3301) for this application.

Reimplemented in **decaf::net::ssl::SSLServerSocketFactory** (p. 3505).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocketFactory.h**

## 6.714 cms::Session Class Reference

A **Session** (p. 3305) object is a single-threaded context for producing and consuming messages.

```
#include <src/main/cms/Session.h>
```

Inheritance diagram for cms::Session:

### Public Types

- enum **AcknowledgeMode** {  
**AUTO\_ACKNOWLEDGE**, **DUPS\_OK\_ACKNOWLEDGE**, **CLIENT\_ACKNOWLEDGE**,  
**SESSION\_TRANSACTED**,  
**INDIVIDUAL\_ACKNOWLEDGE** }

### Public Member Functions

- virtual **~Session** ()
- virtual void **close** ()=0 throw ( CMSEException )  
*Closes this session as well as any active child consumers or producers.*
- virtual void **commit** ()=0 throw ( CMSEException )  
*Commits all messages done in this transaction and releases any locks currently held.*
- virtual void **rollback** ()=0 throw ( CMSEException )  
*Rolls back all messages done in this transaction and releases any locks currently held.*
- virtual void **recover** ()=0 throw ( CMSEException )  
*Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2550) for the specified destination.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination, const std::string &selector)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2550) for the specified destination, using a message selector.*
- virtual **MessageConsumer** \* **createConsumer** (const **Destination** \*destination, const std::string &selector, bool noLocal)=0 throw ( CMSEException )  
*Creates a **MessageConsumer** (p. 2550) for the specified destination, using a message selector.*
- virtual **MessageConsumer** \* **createDurableConsumer** (const **Topic** \*destination, const std::string &name, const std::string &selector, bool noLocal=false)=0 throw ( CMSEException )  
*Creates a durable subscriber to the specified topic, using a **Message** (p. 2493) selector.*

- virtual **MessageProducer** \* **createProducer** (const **Destination** \*destination)=0 throw ( CMSEException )  
*Creates a **MessageProducer** (p. 2681) to send messages to the specified destination.*
- virtual **QueueBrowser** \* **createBrowser** (const cms::Queue \*queue)=0 throw ( CMSEException )  
*Creates a new **QueueBrowser** (p. 3098) to peek at Messages on the given **Queue** (p. 3093).*
- virtual **QueueBrowser** \* **createBrowser** (const cms::Queue \*queue, const std::string &selector)=0 throw ( CMSEException )  
*Creates a new **QueueBrowser** (p. 3098) to peek at Messages on the given **Queue** (p. 3093).*
- virtual **Queue** \* **createQueue** (const std::string &queueName)=0 throw ( CMSEException )  
*Creates a queue identity given a **Queue** (p. 3093) name.*
- virtual **Topic** \* **createTopic** (const std::string &topicName)=0 throw ( CMSEException )  
*Creates a topic identity given a **Queue** (p. 3093) name.*
- virtual **TemporaryQueue** \* **createTemporaryQueue** ()=0 throw ( CMSEException )  
*Creates a **TemporaryQueue** (p. 3701) object.*
- virtual **TemporaryTopic** \* **createTemporaryTopic** ()=0 throw ( CMSEException )  
*Creates a **TemporaryTopic** (p. 3703) object.*
- virtual **Message** \* **createMessage** ()=0 throw ( CMSEException )  
*Creates a new **Message** (p. 2493).*
- virtual **BytesMessage** \* **createBytesMessage** ()=0 throw ( CMSEException )  
*Creates a **BytesMessage** (p. 1023).*
- virtual **BytesMessage** \* **createBytesMessage** (const unsigned char \*bytes, int bytesSize)=0 throw ( CMSEException )  
*Creates a **BytesMessage** (p. 1023) and sets the payload to the passed value.*
- virtual **StreamMessage** \* **createStreamMessage** ()=0 throw ( CMSEException )  
*Creates a new **StreamMessage** (p. 3595).*
- virtual **TextMessage** \* **createTextMessage** ()=0 throw ( CMSEException )  
*Creates a new **TextMessage** (p. 3704).*
- virtual **TextMessage** \* **createTextMessage** (const std::string &text)=0 throw ( CMSEException )  
*Creates a new **TextMessage** (p. 3704) and set the text to the value given.*
- virtual **MapMessage** \* **createMapMessage** ()=0 throw ( CMSEException )  
*Creates a new **MapMessage** (p. 2431).*
- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0 throw ( CMSEException )  
*Returns the acknowledgment mode of the session.*
- virtual bool **isTransacted** () const =0 throw ( CMSEException )  
*Gets if the Sessions is a Transacted **Session** (p. 3305).*
- virtual void **unsubscribe** (const std::string &name)=0 throw ( CMSEException )  
*Unsubscribes a durable subscription that has been created by a client.*

### 6.714.1 Detailed Description

A **Session** (p. 3305) object is a single-threaded context for producing and consuming messages.

A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.
- It is a factory for `TemporaryTopics` and `TemporaryQueues`.
- It provides a way to create **Queue** (p. 3093) or **Topic** (p. 3757) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 2550) until a message arrives. The thread may then use one or more of the Session's `MessageProducers`.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's `close` method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 3305) method that can be called concurrently.
- Invoking any other **Session** (p. 3305) method on a closed session must throw an **IllegalStateException** (p. 1958). Closing a closed session must not throw any exceptions.

### Transacted Sessions

When a **Session** (p. 3305) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 3305) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 3305). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 2681) this implies that all messages sent by the producer are not sent to the Provider unit the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 2550) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all Consumed **Message** (p. 2493) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 2493).

### Since

1.0

## 6.714.2 Member Enumeration Documentation

### 6.714.2.1 enum cms::Session::AcknowledgeMode

#### Enumerator:

**AUTO\_ACKNOWLEDGE** With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.

**DUPS\_OK\_ACKNOWLEDGE** With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

**CLIENT\_ACKNOWLEDGE** With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.

**SESSION\_TRANSACTED** Messages will be consumed when the transaction commits.

**INDIVIDUAL\_ACKNOWLEDGE** **Message** (p. 2493) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

## 6.714.3 Constructor & Destructor Documentation

### 6.714.3.1 virtual cms::Session::~Session ( ) [inline, virtual]

## 6.714.4 Member Function Documentation

6.714.4.1 `virtual void cms::Session::close ( ) throw ( CMSEException ) [pure virtual]`

Closes this session as well as any active child consumers or producers.

### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implements **cms::Closeable** (p. 1120).

Implemented in **activemq::cmsutil::PooledSession** (p. 2907), and **activemq::core::ActiveMQSession** (p. 489).

6.714.4.2 `virtual void cms::Session::commit ( ) throw ( CMSEException ) [pure virtual]`

Commits all messages done in this transaction and releases any locks currently held.

### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
<b>IllegalStateException</b> (p. 1958)	- if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2907), and **activemq::core::ActiveMQSession** (p. 489).

Referenced by `activemq::cmsutil::PooledSession::commit()`.

6.714.4.3 `virtual QueueBrowser* cms::Session::createBrowser ( const cms::Queue * queue ) throw ( CMSEException ) [pure virtual]`

Creates a new **QueueBrowser** (p. 3098) to peek at Messages on the given **Queue** (p. 3093).

### Parameters

<i>queue</i>	the <b>Queue</b> (p. 3093) to browse
--------------	--------------------------------------

### Returns

New **QueueBrowser** (p. 3098) that is owned by the caller.

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2093)	- if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2908), and **activemq::core::ActiveMQSession** (p. 489).

**6.714.4.4** `virtual QueueBrowser* cms::Session::createBrowser ( const cms::Queue  
* queue, const std::string & selector ) throw ( CMSException ) [pure  
virtual]`

Creates a new **QueueBrowser** (p. 3098) to peek at Messages on the given **Queue** (p. 3093).

**Parameters**

<i>queue</i>	the <b>Queue</b> (p. 3093) to browse
<i>selector</i>	the <b>Message</b> (p. 2493) selector to filter which messages are browsed.

**Returns**

New **QueueBrowser** (p. 3098) that is owned by the caller.

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2093)	- if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2908), and **activemq::core::ActiveMQSession** (p. 490).

**6.714.4.5** `virtual BytesMessage* cms::Session::createBytesMessage ( ) throw ( CMSException ) [pure virtual]`

Creates a **BytesMessage** (p. 1023).

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- If an internal error occurs.
---	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2909), and **activemq::core::ActiveMQSession** (p. 490).

6.714.4.6 **virtual BytesMessage\*** cms::Session::createBytesMessage ( **const unsigned char** \* *bytes*, **int** *bytesSize* ) **throw ( CMSEException )** [pure virtual]

Creates a **BytesMessage** (p. 1023) and sets the payload to the passed value.

#### Parameters

<i>bytes</i>	an array of bytes to set in the message
<i>bytesSize</i>	the size of the bytes array, or number of bytes to use

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2909), and **activemq::core::ActiveMQSession** (p. 491).

6.714.4.7 **virtual MessageConsumer\*** cms::Session::createConsumer ( **const Destination** \* *destination* ) **throw ( CMSEException )** [pure virtual]

Creates a **MessageConsumer** (p. 2550) for the specified destination.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1688) that this consumer receiving messages for.
--------------------	---

#### Returns

pointer to a new **MessageConsumer** (p. 2550) that is owned by the caller ( caller deletes )

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
<b>InvalidDestinationException</b> (p. 2093)	- if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2910), and **activemq::core::ActiveMQSession** (p. 492).



6.714.4.8 `virtual MessageConsumer* cms::Session::createConsumer ( const Destination  
* destination, const std::string & selector ) throw ( CMSEException ) [pure  
virtual]`

Creates a **MessageConsumer** (p. 2550) for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1688) that this consumer receiving messages for.
<i>selector</i>	the <b>Message</b> (p. 2493) Selector to use

#### Returns

pointer to a new **MessageConsumer** (p. 2550) that is owned by the caller ( caller deletes )

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
<b>InvalidDestinationException</b> (p. 2093)	- if an invalid destination is specified.
<b>InvalidSelectorException</b> (p. 2099)	- if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2911), and **activemq::core::ActiveMQSession** (p. 492).

6.714.4.9 `virtual MessageConsumer* cms::Session::createConsumer ( const Destination  
* destination, const std::string & selector, bool noLocal ) throw ( CMSEException )  
[pure virtual]`

Creates a **MessageConsumer** (p. 2550) for the specified destination, using a message selector.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1688) that this consumer receiving messages for.
<i>selector</i>	the <b>Message</b> (p. 2493) Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

#### Returns

pointer to a new **MessageConsumer** (p. 2550) that is owned by the caller ( caller deletes )

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2093)	- if an invalid destination is specified.
<b><i>InvalidSelectorException</i></b> (p. 2099)	- if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2911), and **activemq::core::ActiveMQSession** (p. 491).

6.714.4.10 **virtual MessageConsumer\* cms::Session::createDurableConsumer ( const Topic \* *destination*, const std::string & *name*, const std::string & *selector*, bool *noLocal* = false ) throw ( CMSException )** [pure virtual]

Creates a durable subscriber to the specified topic, using a **Message** (p. 2493) selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

**Parameters**

<i>destination</i>	the topic to subscribe to
<i>name</i>	The name used to identify the subscription
<i>selector</i>	the <b>Message</b> (p. 2493) Selector to use
<i>noLocal</i>	if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

**Returns**

pointer to a new durable **MessageConsumer** (p. 2550) that is owned by the caller ( caller deletes )

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- If an internal error occurs.
<b><i>InvalidDestinationException</i></b> (p. 2093)	- if an invalid destination is specified.
<b><i>InvalidSelectorException</i></b> (p. 2099)	- if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2912), and **activemq::core::ActiveMQSession** (p. 492).

6.714.4.11 `virtual MapMessage* cms::Session::createMapMessage ( ) throw ( CMSException ) [pure virtual]`

Creates a new **MapMessage** (p. 2431).

#### Exceptions

<b>CMSException</b> (p. 1130)	- If an internal error occurs.
----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2913), and **activemq::core::ActiveMQSession** (p. 493).

6.714.4.12 `virtual Message* cms::Session::createMessage ( ) throw ( CMSException ) [pure virtual]`

Creates a new **Message** (p. 2493).

#### Exceptions

<b>CMSException</b> (p. 1130)	- If an internal error occurs.
----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2913), and **activemq::core::ActiveMQSession** (p. 493).

6.714.4.13 `virtual MessageProducer* cms::Session::createProducer ( const Destination * destination ) throw ( CMSException ) [pure virtual]`

Creates a **MessageProducer** (p. 2681) to send messages to the specified destination.

#### Parameters

<i>destination</i>	the <b>Destination</b> (p. 1688) to send on
--------------------	---

#### Returns

New **MessageProducer** (p. 2681) that is owned by the caller.

#### Exceptions

<b>CMSException</b> (p. 1130)	- If an internal error occurs.
<b>InvalidDestinationException</b> (p. 2093)	- if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2913), and **activemq::core::ActiveMQSession** (p. 493).

6.714.4.14 `virtual Queue* cms::Session::createQueue ( const std::string & queueName )  
throw ( CMSEException ) [pure virtual]`

Creates a queue identity given a **Queue** (p. 3093) name.

#### Parameters

<i>queueName</i>	the name of the new <b>Queue</b> (p. 3093)
------------------	--

#### Returns

new **Queue** (p. 3093) pointer that is owned by the caller.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2914), and **activemq::core::ActiveMQSession** (p. 494).

6.714.4.15 `virtual StreamMessage* cms::Session::createStreamMessage ( ) throw ( CMSEException ) [pure virtual]`

Creates a new **StreamMessage** (p. 3595).

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2914), and **activemq::core::ActiveMQSession** (p. 494).

6.714.4.16 `virtual TemporaryQueue* cms::Session::createTemporaryQueue ( ) throw ( CMSEException ) [pure virtual]`

Creates a **TemporaryQueue** (p. 3701) object.

#### Returns

new **TemporaryQueue** (p. 3701) pointer that is owned by the caller.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2914), and **activemq::core::ActiveMQSession** (p. 494).

6.714.4.17 **virtual TemporaryTopic\*** cms::Session::createTemporaryTopic ( ) throw ( **CMSEException** ) [pure virtual]

Creates a **TemporaryTopic** (p. 3703) object.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2915), and **activemq::core::ActiveMQSession** (p. 494).

6.714.4.18 **virtual TextMessage\*** cms::Session::createTextMessage ( const std::string & text ) throw ( **CMSEException** ) [pure virtual]

Creates a new **TextMessage** (p. 3704) and set the text to the value given.

#### Parameters

<i>text</i>	the initial text for the message
-------------	----------------------------------

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2915), and **activemq::core::ActiveMQSession** (p. 495).

6.714.4.19 **virtual TextMessage\*** cms::Session::createTextMessage ( ) throw ( **CMSEException** ) [pure virtual]

Creates a new **TextMessage** (p. 3704).

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2915), and **activemq::core::ActiveMQSession** (p. 495).

6.714.4.20 `virtual Topic* cms::Session::createTopic ( const std::string & topicName ) throw ( CMSEException ) [pure virtual]`

Creates a topic identity given a **Queue** (p. 3093) name.

#### Parameters

<i>topicName</i>	the name of the new <b>Topic</b> (p. 3757)
------------------	--

#### Returns

new **Topic** (p. 3757) pointer that is owned by the caller.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2915), and **activemq::core::ActiveMQSession** (p. 495).

6.714.4.21 `virtual AcknowledgeMode cms::Session::getAcknowledgeMode ( ) const throw ( CMSEException ) [pure virtual]`

Returns the acknowledgment mode of the session.

#### Returns

the Sessions Acknowledge Mode

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2916), and **activemq::core::ActiveMQSession** (p. 496).

6.714.4.22 `virtual bool cms::Session::isTransacted ( ) const throw ( CMSEException ) [pure virtual]`

Gets if the Sessions is a Transacted **Session** (p. 3305).

#### Returns

transacted true - false.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2916), and **activemq::core::ActiveMQSession** (p. 499).

**6.714.4.23** `virtual void cms::Session::recover ( ) throw ( CMSEException )` [pure virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "re-delivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if the CMS provider fails to stop and restart message delivery due to some internal error.
<b>IllegalStateException</b> (p. 1958)	- if the method is called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2917), and **activemq::core::ActiveMQSession** (p. 499).

**6.714.4.24** `virtual void cms::Session::rollback ( ) throw ( CMSEException )` [pure virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
<b>IllegalStateException</b> (p. 1958)	- if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2917), and **activemq::core::ActiveMQSession** (p. 501).

6.714.4.25 `virtual void cms::Session::unsubscribe ( const std::string & name ) throw ( CMSEException ) [pure virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active **MessageConsumer** (p. 2550) or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

#### Parameters

<i>name</i>	The name used to identify this subscription
-------------	---

#### Exceptions

<b>CMSEException</b> (p. 1130)	- If an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::cmsutil::PooledSession** (p. 2918), and **activemq::core::ActiveMQSession** (p. 502).

The documentation for this class was generated from the following file:

- `src/main/cms/Session.h`

## 6.715 activemq::cmsutil::SessionCallback Class Reference

Callback for executing any number of operations on a provided CMS Session.

```
#include <src/main/activemq/cmsutil/SessionCallback.h>
```

Inheritance diagram for `activemq::cmsutil::SessionCallback`:

#### Public Member Functions

- `virtual ~SessionCallback ()`
- `virtual void doInCms (cms::Session *session)=0 throw ( cms::CMSEException )`

*Execute any number of operations against the supplied CMS session.*

### 6.715.1 Detailed Description

Callback for executing any number of operations on a provided CMS Session.



## 6.715.2 Constructor & Destructor Documentation

6.715.2.1 `virtual activemq::cmsutil::SessionCallback::~~SessionCallback ( ) [inline, virtual]`

## 6.715.3 Member Function Documentation

6.715.3.1 `virtual void activemq::cmsutil::SessionCallback::doInCms ( cms::Session * session ) throw ( cms::CMSException ) [pure virtual]`

Execute any number of operations against the supplied CMS session.

### Parameters

<i>session</i>	the CMS Session
----------------	-----------------

### Exceptions

<b><i>cms::CMSException</i></b> (p. 1130)	if thrown by CMS API methods
--	------------------------------

Implemented in `activemq::cmsutil::CmsTemplate::ProducerExecutor` (p. 3014), and `activemq::cmsutil::CmsTemplate::ReceiveExecutor` (p. 3120).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionCallback.h`

## 6.716 activemq::commands::SessionId Class Reference

```
#include <src/main/activemq/commands/SessionId.h>
```

Inheritance diagram for `activemq::commands::SessionId`:

### Public Types

- `typedef decaf::lang::PointerComparator< SessionId > COMPARATOR`

### Public Member Functions

- `SessionId ( )`
- `SessionId (const SessionId &other)`
- `SessionId (const ConnectionId *connectionId, long long sessionId)`
- `SessionId (const ProducerId *producerId)`

- **SessionId** (const **ConsumerId** \*consumerId)
- virtual ~**SessionId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **SessionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **ConnectionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **SessionId** &value) const
- virtual bool **equals** (const **SessionId** &value) const
- virtual bool **operator==** (const **SessionId** &value) const
- virtual bool **operator<** (const **SessionId** &value) const
- **SessionId** & **operator=** (const **SessionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_SESSIONID** = 121

### Protected Attributes

- std::string **connectionId**
- long long **value**

### 6.716.1 Member Typedef Documentation

- 6.716.1.1 `typedef decaf::lang::PointerComparator<SessionId>  
activemq::commands::SessionId::COMPARATOR`

### 6.716.2 Constructor & Destructor Documentation

6.716.2.1 `activemq::commands::SessionId::SessionId ( )`

6.716.2.2 `activemq::commands::SessionId::SessionId ( const SessionId & other )`

6.716.2.3 `activemq::commands::SessionId::SessionId ( const ConnectionId * connectionId, long long sessionId )`

6.716.2.4 `activemq::commands::SessionId::SessionId ( const ProducerId * producerId )`

6.716.2.5 `activemq::commands::SessionId::SessionId ( const ConsumerId * consumerId )`

6.716.2.6 `virtual activemq::commands::SessionId::~~SessionId ( ) [virtual]`

### 6.716.3 Member Function Documentation

6.716.3.1 `virtual SessionId* activemq::commands::SessionId::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.716.3.2 `virtual int activemq::commands::SessionId::compareTo ( const SessionId & value ) const [virtual]`

6.716.3.3 `virtual void activemq::commands::SessionId::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.716.3.4 `virtual bool activemq::commands::SessionId::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if `DataSet`'s are Equal.

Implements **activemq::commands::DataSet** (p. 1630).

6.716.3.5 `virtual bool activemq::commands::SessionId::equals ( const SessionId & value ) const [virtual]`

6.716.3.6 `virtual std::string& activemq::commands::SessionId::getConnectionId ( ) [virtual]`

6.716.3.7 `virtual const std::string& activemq::commands::SessionId::getConnectionId ( ) const [virtual]`

6.716.3.8 `virtual unsigned char activemq::commands::SessionId::getDataSetType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataSet** (p. 1628) type copy.

Implements **activemq::commands::DataSet** (p. 1631).

6.716.3.9 `const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId ( ) const`

6.716.3.10 `virtual long long activemq::commands::SessionId::getValue ( ) const [virtual]`

6.716.3.11 `virtual bool activemq::commands::SessionId::operator< ( const SessionId & value ) const [virtual]`

6.716.3.12 `SessionId& activemq::commands::SessionId::operator= ( const SessionId & other )`

6.716.3.13 `virtual bool activemq::commands::SessionId::operator== ( const SessionId & value ) const [virtual]`

6.716.3.14 `virtual void activemq::commands::SessionId::setConnectionId ( const std::string & connectionId ) [virtual]`

## 6.717 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference 3335

---

6.716.3.15 `virtual void activemq::commands::SessionId::setValue ( long long value )`  
[virtual]

6.716.3.16 `virtual std::string activemq::commands::SessionId::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 796).

### 6.716.4 Field Documentation

6.716.4.1 `std::string activemq::commands::SessionId::connectionId`  
[protected]

6.716.4.2 `const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121`  
[static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.716.4.3 `long long activemq::commands::SessionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionId.h`

## 6.717 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3324).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller`:

### Public Member Functions

- **SessionIdMarshaller** ()

- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.717.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3324).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.717.2 Constructor & Destructor Documentation

6.717.2.1 **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::SessionIdMarshaller**  
 ( ) [inline]

6.717.2.2 **virtual activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::~~SessionIdMarshaller**  
 ( ) [inline, virtual]

### 6.717.3 Member Function Documentation

6.717.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::createObject** ( )  
 const [virtual]

Creates a new instance of this marshalable type.

## 6.717 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference 3337

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

**6.717.3.2** virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

**6.717.3.3** virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

**6.717.3.4** virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.717.3.5  virtual int activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.717.3.6  virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).



## 6.718 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference 3339

6.717.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**SessionIdMarshaller.h**

## 6.718 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3328).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.718.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3328).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.718.2 Constructor & Destructor Documentation

6.718.2.1 **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.718.2.2 **virtual activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.718.3 Member Function Documentation

6.718.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.718 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference 3341

6.718.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.718.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.718.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.718.3.5  virtual int activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.718.3.6  virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

## 6.719 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference 3343

6.718.3.7 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**SessionIdMarshaller.h**

## 6.719 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3332).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.719.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3332).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.719.2 Constructor & Destructor Documentation

6.719.2.1 **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.719.2.2 **virtual activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.719.3 Member Function Documentation

6.719.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.719 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller Class Reference 3345

6.719.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.719.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.719.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.719.3.5  virtual int activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.719.3.6  virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).



## 6.720 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference 3347

6.719.3.7 virtual void activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**SessionIdMarshaller.h**

## 6.720 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3336).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.720.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3336).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.720.2 Constructor & Destructor Documentation

6.720.2.1 **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.720.2.2 **virtual activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

## 6.720.3 Member Function Documentation

6.720.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.720 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference 3349

6.720.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.720.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.720.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.720.3.5  virtual int activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.720.3.6  virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

## 6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference 3351

6.720.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**SessionIdMarshaller.h**

## 6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3340).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (OpenWireFormat \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.721.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3340).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.721.2 Constructor & Destructor Documentation

6.721.2.1 **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.721.2.2 **virtual activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

### 6.721.3 Member Function Documentation

6.721.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.721 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference 3353

6.721.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.721.3.3 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.721.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.721.3.5  virtual int activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.721.3.6  virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).



## 6.722 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference 3355

6.721.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightUnmarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataInputStream \* dataIn, utils::BooleanStream \* bs ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**SessionIdMarshaller.h**

## 6.722 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3344).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller:

### Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

## 6.722.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3344).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.722.2 Constructor & Destructor Documentation

6.722.2.1 **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::SessionIdMarshaller**  
( ) [inline]

6.722.2.2 **virtual activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::~~SessionIdMarshaller**  
( ) [inline, virtual]

## 6.722.3 Member Function Documentation

6.722.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

## 6.722 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference 3357

6.722.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.722.3.3 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseMarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.722.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.722.3.5  virtual int activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.722.3.6  virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.722.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionIdMarshaller.h**

## 6.723 activemq::commands::SessionInfo Class Reference

```
#include <src/main/activemq/commands/SessionInfo.h>
```

Inheritance diagram for **activemq::commands::SessionInfo**:

#### Public Member Functions

- **SessionInfo** ()
- virtual **~SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **SessionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*

- virtual bool **equals** (const **DataSet** \*value) const  
*Compares the **DataSet** (p. 1628) passed in to this one, and returns if they are equivalent.*
- unsigned int **getAckMode** () const
- void **setAckMode** (unsigned int mode)
- **Pointer**< **RemoveInfo** > **createRemoveCommand** () const
- virtual const **Pointer**< **SessionId** > & **getSessionId** () const
- virtual **Pointer**< **SessionId** > & **getSessionId** ()
- virtual void **setSessionId** (const **Pointer**< **SessionId** > &sessionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor)  
throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_SESSIONINFO** = 4

### Protected Attributes

- **Pointer**< **SessionId** > **sessionId**

## 6.723.1 Constructor & Destructor Documentation

6.723.1.1 activemq::commands::SessionInfo::SessionInfo ( )

6.723.1.2 virtual activemq::commands::SessionInfo::~~SessionInfo ( ) [virtual]

## 6.723.2 Member Function Documentation

6.723.2.1 virtual **SessionInfo**\* activemq::commands::SessionInfo::cloneDataSet ( )  
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataSet** (p. 1628).

6.723.2.2 `virtual void activemq::commands::SessionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.723.2.3 `Pointer<RemoveInfo> activemq::commands::SessionInfo::createRemoveCommand ( ) const`

6.723.2.4 `virtual bool activemq::commands::SessionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.723.2.5 `unsigned int activemq::commands::SessionInfo::getAckMode ( ) const [inline]`

6.723.2.6 `virtual unsigned char activemq::commands::SessionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.723.2.7 `virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ( ) const [virtual]`

6.723.2.8 `virtual Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ( ) [virtual]`

6.723.2.9 `void activemq::commands::SessionInfo::setAckMode ( unsigned int mode ) [inline]`

6.723.2.10 `virtual void activemq::commands::SessionInfo::setSessionId ( const Pointer<SessionId> & sessionId ) [virtual]`

6.723.2.11 `virtual std::string activemq::commands::SessionInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.723.2.12 `virtual Pointer<Command> activemq::commands::SessionInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

## 6.723.3 Field Documentation

6.723.3.1 `const unsigned char activemq::commands::SessionInfo::ID_SESSIONINFO = 4 [static]`

6.723.3.2 `Pointer<SessionId> activemq::commands::SessionInfo::sessionId [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionInfo.h`



## 6.724 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3352).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.724.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3352).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.724.2 Constructor & Destructor Documentation

6.724.2.1 `activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.724.2.2 `virtual activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.724.3 Member Function Documentation

6.724.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.724.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.724.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

**6.724.3.4** `virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

**6.724.3.5** `virtual int activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.724.3.6  virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.724.3.7  virtual void activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**SessionInfoMarshaller.h**

## 6.725 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3356).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.725.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3356).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.725.2 Constructor & Destructor Documentation

6.725.2.1 `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.725.2.2 `virtual activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.725.3 Member Function Documentation

6.725.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.725.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.725.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

**6.725.3.4** `virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
[virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

**6.725.3.5** `virtual int activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.725.3.6 virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.725.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**SessionInfoMarshaller.h**



## 6.726 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3360).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.726.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3360).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.726.2 Constructor & Destructor Documentation

6.726.2.1 `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.726.2.2 `virtual activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.726.3 Member Function Documentation

6.726.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.726.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.726.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.726.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.726.3.5 virtual int activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.726.3.6  virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.726.3.7  virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionInfoMarshaller.h**

## 6.727 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3364).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.727.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3364).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.727.2 Constructor & Destructor Documentation

6.727.2.1 `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.727.2.2 `virtual activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.727.3 Member Function Documentation

6.727.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.727.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.727.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.727.3.4  virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.727.3.5  virtual int activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.727.3.6  virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.727.3.7  virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**SessionInfoMarshaller.h**



## 6.728 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3368).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.728.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3368).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.728.2 Constructor & Destructor Documentation

6.728.2.1 `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.728.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.728.3 Member Function Documentation

6.728.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.728.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.728.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.728.3.4  virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.728.3.5  virtual int activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.728.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.728.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**SessionInfoMarshaller.h**

## 6.729 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3372).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller:

### Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.729.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3372).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.729.2 Constructor & Destructor Documentation

6.729.2.1 `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::SessionInfoMarshaller ( ) [inline]`

6.729.2.2 `virtual activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::~~SessionInfoMarshaller ( ) [inline, virtual]`

## 6.729.3 Member Function Documentation

6.729.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1578).

6.729.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1585).

6.729.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

**6.729.3.4** virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
[virtual]

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

**6.729.3.5** virtual int activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal1 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
[virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.729.3.6  virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.729.3.7  virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**SessionInfoMarshaller.h**



## 6.730 activemq::cmsutil::SessionPool Class Reference

A pool of CMS sessions from the same connection and with the same acknowledge mode.

```
#include <src/main/activemq/cmsutil/SessionPool.h>
```

### Public Member Functions

- **SessionPool** (**cms::Connection** \*connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** \*resourceLifecycleManager)  
*Constructs a session pool.*
- virtual **~SessionPool** ()  
*Destroys the pooled session objects, but not the underlying session resources.*
- virtual **PooledSession** \* **takeSession** () throw ( cms::CMSException )  
*Takes a session from the pool, creating one if necessary.*
- virtual void **returnSession** (**PooledSession** \*session)  
*Returns a session to the pool.*
- **ResourceLifecycleManager** \* **getResourceLifecycleManager** ()

### Protected Member Functions

- **SessionPool** (const **SessionPool** &)
- **SessionPool** & **operator=** (const **SessionPool** &)

#### 6.730.1 Detailed Description

A pool of CMS sessions from the same connection and with the same acknowledge mode.

Internal session resources are managed through a provided **ResourceLifecycleManager** (p. 3224) , not by this pool. This class is thread-safe.

#### 6.730.2 Constructor & Destructor Documentation

6.730.2.1 **activemq::cmsutil::SessionPool::SessionPool** ( const **SessionPool** & )  
[inline, protected]

6.730.2.2 **activemq::cmsutil::SessionPool::SessionPool** ( **cms::Connection** \* connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** \* resourceLifecycleManager )

Constructs a session pool.

### Parameters

<i>connection</i>	the connection to be used for creating all sessions.
<i>ackMode</i>	the acknowledge mode to be used for all sessions
<i>resourceLifecycleManager</i>	the object responsible for managing the lifecycle of any allocated <b>cms::Session</b> (p. 3305) resources.

6.730.2.3 `virtual activemq::cmsutil::SessionPool::~~SessionPool ( ) [virtual]`

Destroys the pooled session objects, but not the underlying session resources.

That is the job of the **ResourceLifecycleManager** (p. 3224).

### 6.730.3 Member Function Documentation

6.730.3.1 **ResourceLifecycleManager\*** `activemq::cmsutil::SessionPool::getResourceLifecycleManager ( )`  
[inline]

6.730.3.2 **SessionPool&** `activemq::cmsutil::SessionPool::operator= ( const SessionPool & )` [inline, protected]

6.730.3.3 `virtual void activemq::cmsutil::SessionPool::returnSession ( PooledSession * session )` [virtual]

Returns a session to the pool.

#### Parameters

<i>session</i>	the session to be returned.
----------------	-----------------------------

6.730.3.4 `virtual PooledSession* activemq::cmsutil::SessionPool::takeSession ( ) throw ( cms::CMSEException )` [virtual]

Takes a session from the pool, creating one if necessary.

#### Returns

the pooled session object

#### Exceptions

<b>cms::CMSEException</b> (p. 1130)	if an error occurred
--	----------------------

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**SessionPool.h**

## 6.731 activemq::state::SessionState Class Reference

```
#include <src/main/activemq/state/SessionState.h>
```

### Public Member Functions

- **SessionState** (const **Pointer**< **SessionInfo** > &info)
- virtual ~**SessionState** ()
- std::string **toString** () const
- const **Pointer**< **SessionInfo** > **getInfo** () const
- void **addProducer** (const **Pointer**< **ProducerInfo** > &info)
- **Pointer**< **ProducerState** > **removeProducer** (const **Pointer**< **ProducerId** > &id)
- void **addConsumer** (const **Pointer**< **ConsumerInfo** > &info)
- **Pointer**< **ConsumerState** > **removeConsumer** (const **Pointer**< **ConsumerId** > &id)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** () const
- **Pointer**< **ProducerState** > **getProducerState** (const **Pointer**< **ProducerId** > &id)
- std::vector< **Pointer**< **ConsumerState** > > **getConsumerStates** () const
- **Pointer**< **ConsumerState** > **getConsumerState** (const **Pointer**< **ConsumerId** > &id)
- void **checkShutdown** () const
- void **shutdown** ()

### 6.731.1 Constructor & Destructor Documentation

6.731.1.1 **activemq::state::SessionState::SessionState** ( const **Pointer**< **SessionInfo** > &*info* )

6.731.1.2 virtual **activemq::state::SessionState::~~SessionState** ( ) [virtual]

### 6.731.2 Member Function Documentation

6.731.2.1 void **activemq::state::SessionState::addConsumer** ( const **Pointer**< **ConsumerInfo** > &*info* ) [inline]

6.731.2.2 void **activemq::state::SessionState::addProducer** ( const **Pointer**< **ProducerInfo** > &*info* )

- 6.731.2.3 `void activemq::state::SessionState::checkShutdown ( ) const`
- 6.731.2.4 `Pointer<ConsumerState> activemq::state::SessionState::getConsumerState ( const Pointer< ConsumerId > & id ) [inline]`
- 6.731.2.5 `std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates ( ) const [inline]`
- 6.731.2.6 `const Pointer<SessionInfo> activemq::state::SessionState::getInfo ( ) const [inline]`
- 6.731.2.7 `Pointer<ProducerState> activemq::state::SessionState::getProducerState ( const Pointer< ProducerId > & id ) [inline]`
- 6.731.2.8 `std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates ( ) const [inline]`
- 6.731.2.9 `Pointer<ConsumerState> activemq::state::SessionState::removeConsumer ( const Pointer< ConsumerId > & id ) [inline]`
- 6.731.2.10 `Pointer<ProducerState> activemq::state::SessionState::removeProducer ( const Pointer< ProducerId > & id )`
- 6.731.2.11 `void activemq::state::SessionState::shutdown ( ) [inline]`
- 6.731.2.12 `std::string activemq::state::SessionState::toString ( ) const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/SessionState.h`

## 6.732 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

```
#include <src/main/decaf/util/Set.h>
```

Inheritance diagram for `decaf::util::Set< E >`:

### Public Member Functions

- `virtual ~Set ()`

#### 6.732.1 Detailed Description

```
template<typename E>class decaf::util::Set< E >
```

A collection that contains no duplicate elements.

More formally, sets contain no pair of elements  $e_1$  and  $e_2$  such that  $e_1 == e_2$ , and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

#### Since

1.0

### 6.732.2 Constructor & Destructor Documentation

```
6.732.2.1 template<typename E> virtual decaf::util::Set< E >::~~Set ( ) [inline, virtual]
```

The documentation for this class was generated from the following file:

- src/main/decaf/util/Set.h

## 6.733 decaf::lang::Short Class Reference

```
#include <src/main/decaf/lang/Short.h>
```

Inheritance diagram for decaf::lang::Short:

### Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value) throw ( exceptions::NumberFormatException )
- virtual ~**Short** ()
- virtual int **compareTo** (const **Short** &s) const  
*Compares this **Short** (p. 3380) instance with another.*
- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **Short** &s) const

*Compares this object to another and returns true if this object is considered to be less than the one passed.*

- virtual int **compareTo** (const short &s) const

*Compares this **Short** (p. 3380) instance with another.*

- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const

*Compares equality between this object and the one passed.*

- virtual bool **operator<** (const short &s) const

*Compares this object to another and returns true if this object is considered to be less than the one passed.*

- std::string **toString** () const
- virtual double **doubleValue** () const

*Answers the double value which the receiver represents.*

- virtual float **floatValue** () const

*Answers the float value which the receiver represents.*

- virtual unsigned char **byteValue** () const

*Answers the byte value which the receiver represents.*

- virtual short **shortValue** () const

*Answers the short value which the receiver represents.*

- virtual int **intValue** () const

*Answers the int value which the receiver represents.*

- virtual long long **longValue** () const

*Answers the long value which the receiver represents.*

## Static Public Member Functions

- static std::string **toString** (short value)

- static **Short decode** (const std::string &value) throw ( exceptions::NumberFormatException )

*Decodes a **String** (p. 3610) into a **Short** (p. 3380).*

- static short **reverseBytes** (short value)

*Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.*

- static short **parseShort** (const std::string &s, int radix) throw ( exceptions::NumberFormatException )

*Parses the string argument as a signed short in the radix specified by the second argument.*

- static short **parseShort** (const std::string &s) throw ( exceptions::NumberFormatException )

*Parses the string argument as a signed decimal short.*

- static **Short valueOf** (short value)

*Returns a **Short** (p. 3380) instance representing the specified short value.*

- static **Short valueOf** (const std::string &value) throw ( exceptions::NumberFormatException )

Returns a **Short** (p. 3380) object holding the value given by the specified `std::string`.

- static **Short valueOf** (const `std::string` &value, int radix) throw ( `exceptions::NumberFormatException` )

Returns a **Short** (p. 3380) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.

## Static Public Attributes

- static const int **SIZE** = 16  
Size of this objects primitive type in bits.
- static const short **MAX\_VALUE** = (short)0x7FFF  
Max Value for this Object's primitive type.
- static const short **MIN\_VALUE** = (short)0x8000  
Max Value for this Object's primitive type.

## 6.733.1 Constructor & Destructor Documentation

### 6.733.1.1 decaf::lang::Short::Short ( short value )

#### Parameters

<i>value</i>	- short to wrap
--------------	-----------------

### 6.733.1.2 decaf::lang::Short::Short ( const `std::string` & value ) throw ( `exceptions::NumberFormatException` )

#### Parameters

<i>value</i>	- string value to convert to short and wrap
--------------	---

#### Exceptions

<i>NumberFormatException</i>	
------------------------------	--

### 6.733.1.3 virtual decaf::lang::Short::~Short ( ) [inline, virtual]

## 6.733.2 Member Function Documentation

### 6.733.2.1 virtual unsigned char decaf::lang::Short::byteValue ( ) const [inline, virtual]

Answers the byte value which the receiver represents.

**Returns**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2787).

6.733.2.2 `virtual int decaf::lang::Short::compareTo ( const short & s ) const` [virtual]

Compares this **Short** (p. 3380) instance with another.

**Parameters**

<code>s</code>	- the <b>Short</b> (p. 3380) instance to be compared
----------------	--

**Returns**

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **short** > (p. 1187).

6.733.2.3 `virtual int decaf::lang::Short::compareTo ( const Short & s ) const` [virtual]

Compares this **Short** (p. 3380) instance with another.

**Parameters**

<code>s</code>	- the <b>Short</b> (p. 3380) instance to be compared
----------------	--

**Returns**

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **Short** > (p. 1187).

6.733.2.4 `static Short decaf::lang::Short::decode ( const std::string & value ) throw ( exceptions::NumberFormatException )` [static]

Decodes a **String** (p. 3610) into a **Short** (p. 3380).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p. 3386) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified **String** (p. 3610) is the minus sign. No whitespace characters are permitted in the string.



**Parameters**

<i>value</i>	- The string to decode
--------------	------------------------

**Returns**

a **Short** (p. 3380) object containing the decoded value

**Exceptions**

<i>NumberFomatException</i>	if the string is not formatted correctly.
-----------------------------	---

**6.733.2.5** `virtual double decaf::lang::Short::doubleValue ( ) const` `[inline, virtual]`

Answers the double value which the receiver represents.

**Returns**

double the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

**6.733.2.6** `bool decaf::lang::Short::equals ( const Short & s ) const` `[inline, virtual]`

**Returns**

true if the two **Short** (p. 3380) Objects have the same value.

Implements **decaf::lang::Comparable< Short >** (p. 1188).

**6.733.2.7** `bool decaf::lang::Short::equals ( const short & s ) const` `[inline, virtual]`

**Returns**

true if the two **Short** (p. 3380) Objects have the same value.

Implements **decaf::lang::Comparable< short >** (p. 1188).

**6.733.2.8** `virtual float decaf::lang::Short::floatValue ( ) const` `[inline, virtual]`

Answers the float value which the receiver represents.

**Returns**

float the value of the receiver.

Implements **decaf::lang::Number** (p. 2787).

6.733.2.9 `virtual int decaf::lang::Short::intValue ( ) const [inline, virtual]`

Answers the int value which the receiver represents.

#### Returns

int the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.733.2.10 `virtual long long decaf::lang::Short::longValue ( ) const [inline, virtual]`

Answers the long value which the receiver represents.

#### Returns

long the value of the receiver.

Implements **decaf::lang::Number** (p. 2788).

6.733.2.11 `virtual bool decaf::lang::Short::operator< ( const Short & s ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<code>s</code>	- the value to be compared to this one.
----------------	---

#### Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< Short >** (p. 1188).

6.733.2.12 `virtual bool decaf::lang::Short::operator< ( const short & s ) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<code>s</code>	- the value to be compared to this one.
----------------	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **short** > (p. 1188).

6.733.2.13 `virtual bool decaf::lang::Short::operator==( const Short & s ) const [inline, virtual]`

Compares equality between this object and the one passed.

**Parameters**

s	- the value to be compared to this one.
---	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **Short** > (p. 1189).

6.733.2.14 `virtual bool decaf::lang::Short::operator==( const short & s ) const [inline, virtual]`

Compares equality between this object and the one passed.

**Parameters**

s	- the value to be compared to this one.
---	---

**Returns**

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **short** > (p. 1189).

6.733.2.15 `static short decaf::lang::Short::parseShort ( const std::string & s, int radix ) throw ( exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed short in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 1072) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type **NumberFormatException** is thrown if any of the following situations occurs: \* The first argument is null or is a string of length zero. \* The radix is either smaller than **Character.MIN\_RADIX** (p. 1076) or larger than **Character.MAX\_RADIX**

(p. 1076). \* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. \* The value represented by the string is not a value of type short.

#### Parameters

<i>s</i>	- the <b>String</b> (p. 3610) containing the short representation to be parsed
<i>radix</i>	- the radix to be used while parsing <i>s</i>

#### Returns

the short represented by the string argument in the specified radix.

#### Exceptions

<i>NumberFormatException</i>	- If <b>String</b> (p. 3610) does not contain a parsable short.
------------------------------	---

6.733.2.16 `static short decaf::lang::Short::parseShort ( const std::string & s ) throw ( exceptions::NumberFormatException ) [static]`

Parses the string argument as a signed decimal short.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseShort( const std::string, int )` method.

#### Parameters

<i>s</i>	- <b>String</b> (p. 3610) to convert to a short
----------	---

#### Returns

the converted short value

#### Exceptions

<i>NumberFormatException</i>	if the string is not a short.
------------------------------	-------------------------------

6.733.2.17 `static short decaf::lang::Short::reverseBytes ( short value ) [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

#### Parameters

<i>value</i>	- the short whose bytes we are to reverse
--------------	---

**Returns**

the reversed short.

6.733.2.18 `virtual short decaf::lang::Short::shortValue ( ) const [inline, virtual]`

Answers the short value which the receiver represents.

**Returns**

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 2788).

6.733.2.19 `static std::string decaf::lang::Short::toString ( short value ) [static]`

**Returns**

a string representing the primitive value as Base 10

6.733.2.20 `std::string decaf::lang::Short::toString ( ) const`

**Returns**

this **Short** (p. 3380) Object as a **String** (p. 3610) Representation

6.733.2.21 `static Short decaf::lang::Short::valueOf ( short value ) [static]`

Returns a **Short** (p. 3380) instance representing the specified short value.

**Parameters**

<i>value</i>	- the short to wrap
--------------	---------------------

**Returns**

the new **Short** (p. 3380) object wrapping value.

6.733.2.22 `static Short decaf::lang::Short::valueOf ( const std::string & value ) throw ( exceptions::NumberFormatException ) [static]`

Returns a **Short** (p. 3380) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the `parseShort( std::string )` method. The result is a **Short** (p. 3380) object that represents the short value specified by the string.

**Parameters**

<i>value</i>	- std::string to parse as base 10
--------------	-----------------------------------

**Returns**

new **Short** (p. 3380) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a decimal short.
------------------------------	---------------------------------------

6.733.2.23 **static Short decaf::lang::Short::valueOf ( const std::string & value, int radix ) throw ( exceptions::NumberFormatException )** [static]

Returns a **Short** (p. 3380) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the parseShort( std::string, int ) method. The result is a **Short** (p. 3380) object that represents the short value specified by the string.

**Parameters**

<i>value</i>	- std::string to parse as base ( radix )
<i>radix</i>	- base of the string to parse.

**Returns**

new **Short** (p. 3380) Object wrapping the primitive

**Exceptions**

<i>NumberFormatException</i>	if the string is not a valid short.
------------------------------	-------------------------------------

**6.733.3 Field Documentation**

6.733.3.1 **const short decaf::lang::Short::MAX\_VALUE = (short)0x7FFF** [static]

Max Value for this Object's primitive type.

6.733.3.2 **const short decaf::lang::Short::MIN\_VALUE = (short)0x8000** [static]

Max Value for this Object's primitive type.

6.733.3.3 `const int decaf::lang::Short::SIZE = 16` [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Short.h`

## 6.734 decaf::internal::nio::ShortArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/ShortArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

### Public Member Functions

- **ShortArrayBuffer** (int size, bool readOnly=false) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ShortArrayBuffer** (p. 3390) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ShortArrayBuffer** (short \*array, int size, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a **ShortArrayBuffer** (p. 3390) object that wraps the given array.*
- **ShortArrayBuffer** (const decaf::lang::Pointer< **ByteArrayAdapter** > &array, int offset, int length, bool readOnly=false) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.*
- **ShortArrayBuffer** (const **ShortArrayBuffer** &other)  
*Create a **ShortArrayBuffer** (p. 3390) that mirrors this one, meaning it shares a reference to this buffers ByteArrayAdapter and when changes are made to that data it is reflected in both.*
- virtual ~**ShortArrayBuffer** ()
- virtual short \* **array** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )  
*Returns the short array that backs this buffer (optional operation).  
Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.  
Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.*  
**Returns**  
*the array that backs this **Buffer** (p. 887)*  
**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual int **arrayOffset** () throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

The offset into the backing array where index zero starts.

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
UnsupportedOperationException	if the underlying store has no array.

- virtual ShortBuffer \* **asReadOnlyBuffer** () const

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only short buffer which the caller then owns.

- virtual ShortBuffer & **compact** () throw ( decaf::nio::ReadOnlyBufferException )

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **ShortBuffer** (p. 3401).

#### Exceptions

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------



- virtual ShortBuffer \* **duplicate** ()

*Creates a new short buffer that shares this buffer's content.*

*The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*a new short **Buffer** (p. 887) which the caller owns.*

- virtual short **get** () throw ( decaf::nio::BufferUnderflowException )

*Relative get method.*

*Reads the value at this buffer's current position, and then increments the position.*

**Returns**

*the short at the current position.*

**Exceptions**

<b>BufferUnderflowException</b> (p. 916)	<i>if there no more data to return.</i>
---	---

- virtual short **get** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Absolute get method.*

*Reads the value at the given index.*

**Parameters**

index	<i>The index in the <b>Buffer</b> (p. 887) where the short is to be read.</i>
-------	---

**Returns**

*the short that is located at the given index.*

**Exceptions**

<b>IndexOutOfBoundsException</b>	<i>if index is not smaller than the buffer's limit, or the index is negative.</i>
----------------------------------	---

- virtual bool **hasArray** () const

*Tells whether or not this buffer is backed by an accessible short array.*

*If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.*

**Returns**

*true if, and only if, this buffer is backed by an array and is not read-only.*

- virtual bool **isReadOnly** () const

*Tells whether or not this buffer is read-only.*

**Returns**

*true if, and only if, this buffer is read-only.*

- virtual ShortBuffer & **put** (short value) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )

*Writes the given shorts into this buffer at the current position, and then increments the position.*

**Parameters**

value	<i>The shorts value to be written.</i>
-------	--

**Returns**

*a reference to this buffer.*

**Exceptions**

<b>BufferOverflowException</b> (p. 914)	<i>if this buffer's current position is not smaller than its limit.</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only.</i>

- virtual ShortBuffer & **put** (int index, short value) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )

*Writes the given shorts into this buffer at the given index.*

**Parameters**

index	<i>The position in the <b>Buffer</b> (p. 887) to write the data.</i>
value	<i>The shorts to write.</i>

**Returns**

*a reference to this buffer.*

**Exceptions**

IndexOutOfBoundsException	<i>if index greater than the buffer's limit minus the size of the type being written.</i>
<b>ReadOnlyBufferException</b> (p. 3115)	<i>if this buffer is read-only.</i>

- virtual ShortBuffer \* **slice** () const

*Creates a new **ShortBuffer** (p. 3401) whose content is a shared subsequence of this buffer's content.*

*The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.*

*The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.*

**Returns**

*the newly create **ShortBuffer** (p. 3401) which the caller owns.*

**Protected Member Functions**

- virtual void **setReadOnly** (bool value)

*Sets this **ShortArrayBuffer** (p. 3390) as Read-Only.*

**6.734.1 Constructor & Destructor Documentation**

6.734.1.1 `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer ( int size, bool readOnly = false ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Creates a **ShortArrayBuffer** (p. 3390) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>size</i>	The size of the array, this is the limit we read and write to.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>IllegalArgumentException</i>	if the capacity value is negative.
---------------------------------	------------------------------------

6.734.1.2 `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer ( short * array, int size, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a **ShortArrayBuffer** (p. 3390) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

#### Parameters

<i>array</i>	The actual array to wrap.
<i>size</i>	The size of the given array.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if buffer is NULL
<i>IndexOutOfBoundsException</i>	if offset is greater than array capacity.

6.734.1.3 `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer ( const decaf::lang::Pointer< ByteArrayAdapter > & array, int offset, int length, bool readOnly = false ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )`

Creates a byte buffer that wraps the passed ByteArrayAdapter and start at the given offset.

The capacity and limit of the new **ShortArrayBuffer** (p. 3390) will be that of the remain-

ing capacity of the passed buffer.

#### Parameters

<i>array</i>	The ByteAdapter to wrap.
<i>offset</i>	The position that is this buffers start position.
<i>length</i>	The limit of how many bytes into the array this Buffer can write.
<i>readOnly</i>	Boolean indicating if this buffer should be read-only, default as false.

#### Exceptions

<i>NullPointerException</i>	if array is NULL
<i>IndexOutOfBoundsException</i>	if offset + length is greater than array size.

6.734.1.4 `decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer ( const ShortArrayBuffer & other )`

Create a **ShortArrayBuffer** (p. 3390) that mirrors this one, meaning it shares a reference to this buffers ByteAdapter and when changes are made to that data it is reflected in both.

#### Parameters

<i>other</i>	The <b>ShortArrayBuffer</b> (p. 3390) this one is to mirror.
--------------	--

6.734.1.5 `virtual decaf::internal::nio::ShortArrayBuffer::~~ShortArrayBuffer ( )`  
[virtual]

### 6.734.2 Member Function Documentation

6.734.2.1 `virtual short* decaf::internal::nio::ShortArrayBuffer::array ( ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException )` [virtual]

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

#### Returns

the array that backs this **Buffer** (p. 887)

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3404).

6.734.2.2 virtual int decaf::internal::nio::ShortArrayBuffer::arrayOffset ( ) throw  
( decaf::lang::exceptions::UnsupportedOperationException,  
decaf::nio::ReadOnlyBufferException ) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

### Returns

The offset into the backing array where index zero starts.

### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 3404).

6.734.2.3 virtual ShortBuffer\* decaf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer ( )  
const [virtual]

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

**Returns**

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 3405).

6.734.2.4 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact ( ) throw ( decaf::nio::ReadOnlyBufferException )` [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index  $p = \text{position}()$  (p. 892) is copied to index zero, the byte at index  $p + 1$  is copied to index one, and so forth until the byte at index  $\text{limit}()$  (p. 891) - 1 is copied to index  $n = \text{limit}()$  (p. 891) - 1 -  $p$ . The buffer's position is then set to  $n+1$  and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

**Returns**

a reference to this **ShortBuffer** (p. 3401).

**Exceptions**

<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.
---	------------------------------

Implements **decaf::nio::ShortBuffer** (p. 3405).

6.734.2.5 `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::duplicate ( )` [virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

**Returns**

a new short **Buffer** (p. 887) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3406).

6.734.2.6 virtual short decaf::internal::nio::ShortArrayBuffer::get ( ) throw ( decaf::nio::BufferUnderflowException ) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

#### Returns

the short at the current position.

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there no more data to return.
---	----------------------------------

Implements **decaf::nio::ShortBuffer** (p. 3408).

6.734.2.7 virtual short decaf::internal::nio::ShortArrayBuffer::get ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]

Absolute get method.

Reads the value at the given index.

#### Parameters

index	The index in the <b>Buffer</b> (p. 887) where the short is to be read.
-------	--

#### Returns

the short that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or the index is negative.
----------------------------------	--

Implements **decaf::nio::ShortBuffer** (p. 3406).

6.734.2.8 virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray ( ) const [inline, virtual]

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

**Returns**

true if, and only if, this buffer is backed by an array and is not read-only.

Implements **decaf::nio::ShortBuffer** (p. 3408).

6.734.2.9 `virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly ( ) const`  
`[inline, virtual]`

Tells whether or not this buffer is read-only.

**Returns**

true if, and only if, this buffer is read-only.

Implements **decaf::nio::Buffer** (p. 890).

6.734.2.10 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put ( int index, short value ) throw ( lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException )` `[virtual]`

Writes the given shorts into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The shorts to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3409).

6.734.2.11 `virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put ( short value ) throw ( decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException )` `[virtual]`

Writes the given shorts into this buffer at the current position, and then increments the position.



### Parameters

<i>value</i>	The shorts value to be written.
--------------	---------------------------------

### Returns

a reference to this buffer.

### Exceptions

<b>BufferOverflowException</b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

Implements **decaf::nio::ShortBuffer** (p. 3408).

6.734.2.12 virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly ( bool *value* )  
[inline, protected, virtual]

Sets this **ShortArrayBuffer** (p. 3390) as Read-Only.

### Parameters

<i>value</i>	Boolean value, true if this buffer is to be read-only, false otherwise.
--------------	---

6.734.2.13 virtual ShortBuffer\* decaf::internal::nio::ShortArrayBuffer::slice ( ) const  
[virtual]

Creates a new **ShortBuffer** (p. 3401) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

the newly create **ShortBuffer** (p. 3401) which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 3411).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ShortArrayBuffer.h**

## 6.735 decaf::nio::ShortBuffer Class Reference

This class defines four categories of operations upon short buffers:

```
#include <src/main/decaf/nio/ShortBuffer.h>
```

Inheritance diagram for decaf::nio::ShortBuffer:

### Public Member Functions

- virtual **~ShortBuffer** ()
- virtual std::string **toString** () const
- virtual short \* **array** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the short array that backs this buffer (optional operation).*
- virtual int **arrayOffset** ()=0 throw ( decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException )  
*Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).*
- virtual **ShortBuffer** \* **asReadOnlyBuffer** () const =0  
*Creates a new, read-only short buffer that shares this buffer's content.*
- virtual **ShortBuffer** & **compact** ()=0 throw ( ReadOnlyBufferException )  
*Compacts this buffer.*
- virtual **ShortBuffer** \* **duplicate** ()=0  
*Creates a new short buffer that shares this buffer's content.*
- virtual short **get** ()=0 throw ( BufferUnderflowException )  
*Relative get method.*
- virtual short **get** (int index) const =0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException )  
*Absolute get method.*
- **ShortBuffer** & **get** (std::vector< short > buffer) throw ( BufferUnderflowException )  
*Relative bulk get method.*
- **ShortBuffer** & **get** (short \*buffer, int size, int offset, int length) throw ( BufferUnderflowException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*Relative bulk get method.*
- virtual bool **hasArray** () const =0  
*Tells whether or not this buffer is backed by an accessible short array.*
- **ShortBuffer** & **put** (**ShortBuffer** &src) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IllegalArgumentException )  
*This method transfers the shorts remaining in the given source buffer into this buffer.*

- **ShortBuffer** & **put** (const short \*buffer, int size, int offset, int length) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )  
*This method transfers shorts into this buffer from the given source array.*
- **ShortBuffer** & **put** (std::vector< short > &buffer) throw ( BufferOverflowException, ReadOnlyBufferException )  
*This method transfers the entire content of the given source shorts array into this buffer.*
- virtual **ShortBuffer** & **put** (short value)=0 throw ( BufferOverflowException, ReadOnlyBufferException )  
*Writes the given shorts into this buffer at the current position, and then increments the position.*
- virtual **ShortBuffer** & **put** (int index, short value)=0 throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )  
*Writes the given shorts into this buffer at the given index.*
- virtual **ShortBuffer** \* **slice** () const =0  
*Creates a new **ShortBuffer** (p. 3401) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **ShortBuffer** &value) const
- virtual bool **equals** (const **ShortBuffer** &value) const
- virtual bool **operator==** (const **ShortBuffer** &value) const
- virtual bool **operator<** (const **ShortBuffer** &value) const

### Static Public Member Functions

- static **ShortBuffer** \* **allocate** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Allocates a new Double buffer.*
- static **ShortBuffer** \* **wrap** (short \*array, int size, int offset, int length) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Wraps the passed buffer with a new **ShortBuffer** (p. 3401).*
- static **ShortBuffer** \* **wrap** (std::vector< short > &buffer)  
*Wraps the passed STL short Vector in a **ShortBuffer** (p. 3401).*

### Protected Member Functions

- **ShortBuffer** (int capacity) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Creates a **ShortBuffer** (p. 3401) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

### 6.735.1 Detailed Description

This class defines four categories of operations upon short buffers:

- o Absolute and relative get and put methods that read and write single shorts;
- o Relative bulk get methods that transfer contiguous sequences of shorts from this buffer into an array;
- and
- o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer
- o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

### 6.735.2 Constructor & Destructor Documentation

**6.735.2.1** `decaf::nio::ShortBuffer::ShortBuffer ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [protected]

Creates a **ShortBuffer** (p.3401) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

#### Parameters

<i>capacity</i>	The size and limit of the <b>Buffer</b> (p.887) in doubles
-----------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if capacity is negative.
---------------------------------	--------------------------

**6.735.2.2** `virtual decaf::nio::ShortBuffer::~~ShortBuffer ( )` [inline, virtual]

### 6.735.3 Member Function Documentation

**6.735.3.1** `static ShortBuffer* decaf::nio::ShortBuffer::allocate ( int capacity ) throw ( decaf::lang::exceptions::IllegalArgumentException )` [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

#### Parameters

<i>capacity</i>	The size of the Double buffer in shorts.
-----------------	--

**Returns**

the **ShortBuffer** (p. 3401) that was allocated, caller owns.

```
6.735.3.2  virtual short* decaf::nio::ShortBuffer::array ( ) throw (
            decaf::lang::exceptions::UnsupportedOperationException,
            ReadOnlyBufferException ) [pure virtual]
```

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

the array that backs this **Buffer** (p. 887)

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3395).

```
6.735.3.3  virtual int decaf::nio::ShortBuffer::arrayOffset ( ) throw (
            decaf::lang::exceptions::UnsupportedOperationException,
            ReadOnlyBufferException ) [pure virtual]
```

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

**Returns**

The offset into the backing array where index zero starts.

**Exceptions**

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this <b>Buffer</b> (p. 887) is read only.
<b><i>UnsupportedOperationException</i></b>	if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3396).

**6.735.3.4** `virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer ( ) const`  
`[pure virtual]`

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

#### Returns

The new, read-only short buffer which the caller then owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3396).

**6.735.3.5** `virtual ShortBuffer& decaf::nio::ShortBuffer::compact ( ) throw (`  
`ReadOnlyBufferException ) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 892) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 891) - 1 is copied to index `n = limit()` (p. 891) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

#### Returns

a reference to this **ShortBuffer** (p. 3401).

#### Exceptions

<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.
--	------------------------------

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3397).

6.735.3.6 `virtual int decaf::nio::ShortBuffer::compareTo ( const ShortBuffer & value ) const`  
[virtual]

6.735.3.7 `virtual ShortBuffer* decaf::nio::ShortBuffer::duplicate ( )` [pure  
virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

### Returns

a new short **Buffer** (p. 887) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3397).

6.735.3.8 `virtual bool decaf::nio::ShortBuffer::equals ( const ShortBuffer & value ) const`  
[virtual]

6.735.3.9 `ShortBuffer& decaf::nio::ShortBuffer::get ( std::vector< short > buffer ) throw (`  
`BufferUnderflowException )`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize( N )` before calling this get method.

### Returns

a reference to this **Buffer** (p. 887).

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there are fewer than length shorts remaining in this buffer.
--	---

6.735.3.10 `virtual short decaf::nio::ShortBuffer::get ( int index ) const throw (`  
`decaf::lang::exceptions::IndexOutOfBoundsException )` [pure  
virtual]

Absolute get method.

Reads the value at the given index.

#### Parameters

<i>index</i>	The index in the <b>Buffer</b> (p. 887) where the short is to be read.
--------------	--

#### Returns

the short that is located at the given index.

#### Exceptions

<i>IndexOutOfBoundsException</i>	if index is not smaller than the buffer's limit, or the index is negative.
----------------------------------	--

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3398).

6.735.3.11 **ShortBuffer& decaf::nio::ShortBuffer::get ( short \* *buffer*, int *size*, int *offset*, int *length* ) throw ( **BufferUnderflowException**, **decaf::lang::exceptions::IndexOutOfBoundsException**, **decaf::lang::exceptions::NullPointerException** )**

Relative bulk get method.

This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if *length* > **remaining()** (p. 892), then no bytes are transferred and a **BufferUnderflowException** (p. 916) is thrown.

Otherwise, this method copies *length* shorts from this buffer into the given array, starting at the current position of this buffer and at the given *offset* in the array. The position of this buffer is then incremented by *length*.

#### Parameters

<i>buffer</i>	The pointer to an allocated buffer to fill.
<i>size</i>	The size of the buffer provided.
<i>offset</i>	The position in the buffer to start filling.
<i>length</i>	The amount of data to put in the passed buffer.

#### Returns

a reference to this **Buffer** (p. 887).

#### Exceptions

<b>BufferUnderflowException</b> (p. 916)	if there are fewer than <i>length</i> shorts remaining in this buffer
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of <i>size</i> , <i>offset</i> , or <i>length</i> are not met.



6.735.3.12 `virtual short decaf::nio::ShortBuffer::get ( ) throw ( BufferUnderflowException ) [pure virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

### Returns

the short at the current position.

### Exceptions

<b><i>BufferUnderflowException</i></b> (p. 916)	if there no more data to return.
--	----------------------------------

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3398).

6.735.3.13 `virtual bool decaf::nio::ShortBuffer::hasArray ( ) const [pure virtual]`

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

### Returns

true if, and only if, this buffer is backed by an array and is not read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3398).

6.735.3.14 `virtual bool decaf::nio::ShortBuffer::operator< ( const ShortBuffer & value ) const [virtual]`

6.735.3.15 `virtual bool decaf::nio::ShortBuffer::operator== ( const ShortBuffer & value ) const [virtual]`

6.735.3.16 `virtual ShortBuffer& decaf::nio::ShortBuffer::put ( short value ) throw ( BufferOverflowException, ReadOnlyBufferException ) [pure virtual]`

Writes the given shorts into this buffer at the current position, and then increments the position.

### Parameters

<i>value</i>	The shorts value to be written.
--------------	---------------------------------

**Returns**

a reference to this buffer.

**Exceptions**

<b><i>BufferOverflowException</i></b> (p. 914)	if this buffer's current position is not smaller than its limit.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3399).

6.735.3.17 **virtual ShortBuffer& decaf::nio::ShortBuffer::put ( int *index*, short *value* ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException )** [pure virtual]

Writes the given shorts into this buffer at the given index.

**Parameters**

<i>index</i>	The position in the <b>Buffer</b> (p. 887) to write the data.
<i>value</i>	The shorts to write.

**Returns**

a reference to this buffer.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index greater than the buffer's limit minus the size of the type being written.
<b><i>ReadOnlyBufferException</i></b> (p. 3115)	if this buffer is read-only.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3399).

6.735.3.18 **ShortBuffer& decaf::nio::ShortBuffer::put ( const short \* *buffer*, int *size*, int *offset*, int *length* ) throw ( BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )**

This method transfers shorts into this buffer from the given source array.

If there are more shorts to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 892), then no shorts are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

#### Parameters

<i>buffer</i>	The array from which shorts are to be read.
<i>size</i>	The size of the buffer passed.
<i>offset</i>	The offset within the array of the first char to be read.
<i>length</i>	The number of shorts to be read from the given array.

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only
<i>NullPointerException</i>	if the passed buffer is null.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

#### 6.735.3.19 ShortBuffer& decaf::nio::ShortBuffer::put ( std::vector< short > & buffer ) throw ( BufferOverflowException, ReadOnlyBufferException )

This method transfers the entire content of the given source shorts array into this buffer.

This is the same as calling put( &buffer[0], 0, buffer.size()).

#### Parameters

<i>buffer</i>	The buffer whose contents are copied to this <b>ShortBuffer</b> (p. 3401).
---------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

6.735.3.20 **ShortBuffer& decaf::nio::ShortBuffer::put ( ShortBuffer & src )**  
**throw ( BufferOverflowException, ReadOnlyBufferException,**  
**decaf::lang::exceptions::IllegalArgumentException )**

This method transfers the shorts remaining in the given source buffer into this buffer.

If there are more shorts remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 892), then no shorts are transferred and a **BufferOverflowException** (p. 914) is thrown.

Otherwise, this method copies `n = src.remaining()` shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

#### Parameters

<code>src</code>	The buffer to take shorts from an place in this one.
------------------	--

#### Returns

a reference to this buffer.

#### Exceptions

<b>BufferOverflowException</b> (p. 914)	if there is insufficient space in this buffer for the remaining shorts in the source buffer.
<b>IllegalArgumentException</b>	if the source buffer is this buffer.
<b>ReadOnlyBufferException</b> (p. 3115)	if this buffer is read-only.

6.735.3.21 **virtual ShortBuffer\* decaf::nio::ShortBuffer::slice ( ) const** [pure virtual]

Creates a new **ShortBuffer** (p. 3401) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

#### Returns

the newly create **ShortBuffer** (p. 3401) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 3400).

6.735.3.22 `virtual std::string decaf::nio::ShortBuffer::toString ( ) const [virtual]`

#### Returns

a std::string describing this object

6.735.3.23 `static ShortBuffer* decaf::nio::ShortBuffer::wrap ( short * array, int size, int offset, int length ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [static]`

Wraps the passed buffer with a new **ShortBuffer** (p. 3401).

The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>array</i>	The array that will back the new buffer.
<i>size</i>	The size of the passed in array.
<i>offset</i>	The offset of the subarray to be used.
<i>length</i>	The length of the subarray to be used.

#### Returns

a new **ShortBuffer** (p. 3401) that is backed by buffer, caller owns.

#### Exceptions

<i>NullPointerException</i>	if the array pointer is NULL.
<i>IndexOutOfBoundsException</i>	if the preconditions of size, offset, or length are not met.

6.735.3.24 `static ShortBuffer* decaf::nio::ShortBuffer::wrap ( std::vector< short > & buffer ) [static]`

Wraps the passed STL short Vector in a **ShortBuffer** (p. 3401).

The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

#### Parameters

<i>buffer</i>	The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize( N ).
---------------	--

**Returns**

a new **ShortBuffer** (p. 3401) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ShortBuffer.h**

**6.736 activemq::commands::ShutdownInfo Class Reference**

```
#include <src/main/activemq/commands/ShutdownInfo.h>
```

Inheritance diagram for activemq::commands::ShutdownInfo:

**Public Member Functions**

- **ShutdownInfo** ()
- virtual **~ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **ShutdownInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual bool **isShutdownInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

**Static Public Attributes**

- static const unsigned char **ID\_SHUTDOWNINFO** = 11

## 6.736.1 Constructor & Destructor Documentation

6.736.1.1 `activemq::commands::ShutdownInfo::ShutdownInfo ( )`

6.736.1.2 `virtual activemq::commands::ShutdownInfo::~~ShutdownInfo ( ) [virtual]`

## 6.736.2 Member Function Documentation

6.736.2.1 `virtual ShutdownInfo* activemq::commands::ShutdownInfo::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.736.2.2 `virtual void activemq::commands::ShutdownInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

6.736.2.3 `virtual bool activemq::commands::ShutdownInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.736.2.4 `virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.736.2.5 `virtual bool activemq::commands::ShutdownInfo::isShutdownInfo ( ) const [inline, virtual]`

#### Returns

an answer of true to the **isShutdownInfo()** (p. 3415) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 728).

6.736.2.6 `virtual std::string activemq::commands::ShutdownInfo::toString ( ) const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.736.2.7 `virtual Pointer<Command> activemq::commands::ShutdownInfo::visit ( activemq::state::CommandVisitor * visitor ) throw ( exceptions::ActiveMQException ) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).



### 6.736.3 Field Documentation

6.736.3.1 `const unsigned char activemq::commands::ShutdownInfo::ID_-  
SHUTDOWNINFO = 11` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ShutdownInfo.h`

## 6.737 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3416).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller`:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**)  
throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.737.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3416).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.737.2 Constructor & Destructor Documentation

6.737.2.1 **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

6.737.2.2 **virtual activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.737.3 Member Function Documentation

6.737.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.737.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.737.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.737.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

6.737.3.5 `virtual int activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.737.3.6  virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.737.3.7  virtual void activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

## 6.738 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller

### Class Reference 3431

---

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**ShutdownInfoMarshaller.h**

## 6.738 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3420).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**:

#### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.738.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3420).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.738.2 Constructor & Destructor Documentation

6.738.2.1 **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

6.738.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.738.3 Member Function Documentation

6.738.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.738.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

## 6.738 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller

### Class Reference 3433

6.738.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

6.738.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

6.738.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.738.3.6  virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.738.3.7  virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.739 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller

### Class Reference 3435

---

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ShutdownInfoMarshaller.h**

## 6.739 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3424).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**:

#### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.739.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3424).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.739.2 Constructor & Destructor Documentation

6.739.2.1 **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

6.739.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.739.3 Member Function Documentation

6.739.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.739.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.739.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

6.739.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

6.739.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.739.3.6  virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.739.3.7  virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ShutdownInfoMarshaller.h**

## 6.740 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3428).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller:

### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual ~**ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure** \* **createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.740.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3428).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.740.2 Constructor & Destructor Documentation

6.740.2.1 **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

6.740.2.2 **virtual activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.740.3 Member Function Documentation

6.740.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.740.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.740.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

6.740.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

6.740.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

```
6.740.3.6  virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.740.3.7  virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



## 6.741 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller

### Class Reference 3443

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ShutdownInfoMarshaller.h**

## 6.741 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3432).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**:

#### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.741.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3432).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.741.2 Constructor & Destructor Documentation

6.741.2.1 **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

6.741.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.741.3 Member Function Documentation

6.741.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.741.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.741.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

6.741.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

6.741.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.741.3.6  virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.741.3.7  virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--

## 6.742 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller

### Class Reference 3447

<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ShutdownInfoMarshaller.h**

## 6.742 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3436).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**:

#### Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Un-marshall an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )

*Write a object instance to data output stream.*

### 6.742.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3436).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.742.2 Constructor & Destructor Documentation

6.742.2.1 **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::ShutdownInfoMarshaller**  
( ) [inline]

6.742.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller**  
( ) [inline, virtual]

### 6.742.3 Member Function Documentation

6.742.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::createObject** ( )  
**const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.742.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::getDataStructureType**  
( ) **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.742.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

6.742.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )` [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

6.742.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.742.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.742.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
-------------------	--



<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ShutdownInfoMarshaller.h**

**6.743 decaf::security::SignatureException Class Reference**

```
#include <src/main/decaf/security/SignatureException.h>
```

Inheritance diagram for decaf::security::SignatureException:

**Public Member Functions**

- **SignatureException** () throw ()  
*Default Constructor.*
- **SignatureException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **SignatureException** (const **SignatureException** &ex) throw ()  
*Copy Constructor.*
- **SignatureException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SignatureException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **SignatureException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SignatureException** \* **clone** () const  
*Clones this exception.*
- virtual ~**SignatureException** () throw ()

### 6.743.1 Constructor & Destructor Documentation

6.743.1.1 `decaf::security::SignatureException::SignatureException ( ) throw () [inline]`

Default Constructor.

6.743.1.2 `decaf::security::SignatureException::SignatureException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.743.1.3 `decaf::security::SignatureException::SignatureException ( const SignatureException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.743.1.4 `decaf::security::SignatureException::SignatureException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.743.1.5 `decaf::security::SignatureException::SignatureException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.743.1.6** `decaf::security::SignatureException::SignatureException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	name where exception occurs
<i>lineNumber</i>	line number where the exception occurred.
<i>msg</i>	message to report
<i>...</i>	list of primitives that are formatted into the message

**6.743.1.7** `virtual decaf::security::SignatureException::~~SignatureException ( ) throw ()`  
`[inline, virtual]`

**6.743.2 Member Function Documentation**

**6.743.2.1** `virtual SignatureException* decaf::security::SignatureException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

**Returns**

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1936).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SignatureException.h`

**6.744 decaf::util::logging::SimpleFormatter Class Reference**

Print a brief summary of the **LogRecord** (p. 2370) in a human readable format.

```
#include <src/main/decaf/util/logging/SimpleFormatter.h>
```

Inheritance diagram for `decaf::util::logging::SimpleFormatter`:

## Public Member Functions

- **SimpleFormatter** ()
- virtual **~SimpleFormatter** ()
- virtual `std::string format` (const **LogRecord** &record) const  
*Format the given log record and return the formatted string.*

### 6.744.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 2370) in a human readable format.

The summary will typically be 1 or 2 lines.

#### Since

1.0

### 6.744.2 Constructor & Destructor Documentation

6.744.2.1 `decaf::util::logging::SimpleFormatter::SimpleFormatter ( )`

6.744.2.2 `virtual decaf::util::logging::SimpleFormatter::~~SimpleFormatter ( )` [virtual]

### 6.744.3 Member Function Documentation

6.744.3.1 `virtual std::string decaf::util::logging::SimpleFormatter::format ( const LogRecord & record ) const` [virtual]

Format the given log record and return the formatted string.

#### Parameters

<i>record</i>	The Log Record to Format.
---------------	---------------------------

Implements **decaf::util::logging::Formatter** (p. 1928).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleFormatter.h`

## 6.745 decaf::util::logging::SimpleLogger Class Reference

```
#include <src/main/decaf/util/logging/SimpleLogger.h>
```

### Public Member Functions

- **SimpleLogger** (const std::string &name)  
*Constructor.*
- virtual ~**SimpleLogger** ()  
*Destructor.*
- virtual void **mark** (const std::string &message)  
*Log a Mark Block **Level** (p. 2290) Log.*
- virtual void **debug** (const std::string &file, const int line, const std::string &message)  
*Log a Debug **Level** (p. 2290) Log.*
- virtual void **info** (const std::string &file, const int line, const std::string &message)  
*Log a Informational **Level** (p. 2290) Log.*
- virtual void **warn** (const std::string &file, const int line, const std::string &message)  
*Log a Warning **Level** (p. 2290) Log.*
- virtual void **error** (const std::string &file, const int line, const std::string &message)  
*Log a Error **Level** (p. 2290) Log.*
- virtual void **fatal** (const std::string &file, const int line, const std::string &message)  
*Log a Fatal **Level** (p. 2290) Log.*
- virtual void **log** (const std::string &message)  
*No-frills log.*

### 6.745.1 Constructor & Destructor Documentation

#### 6.745.1.1 decaf::util::logging::SimpleLogger::SimpleLogger ( const std::string & name )

Constructor.

#### 6.745.1.2 virtual decaf::util::logging::SimpleLogger::~SimpleLogger ( ) [virtual]

Destructor.

## 6.745.2 Member Function Documentation

6.745.2.1 `virtual void decaf::util::logging::SimpleLogger::debug ( const std::string & file, const int line, const std::string & message )` [virtual]

Log a Debug **Level** (p. 2290) Log.

6.745.2.2 `virtual void decaf::util::logging::SimpleLogger::error ( const std::string & file, const int line, const std::string & message )` [virtual]

Log a Error **Level** (p. 2290) Log.

6.745.2.3 `virtual void decaf::util::logging::SimpleLogger::fatal ( const std::string & file, const int line, const std::string & message )` [virtual]

Log a Fatal **Level** (p. 2290) Log.

6.745.2.4 `virtual void decaf::util::logging::SimpleLogger::info ( const std::string & file, const int line, const std::string & message )` [virtual]

Log a Informational **Level** (p. 2290) Log.

6.745.2.5 `virtual void decaf::util::logging::SimpleLogger::log ( const std::string & message )` [virtual]

No-frills log.

6.745.2.6 `virtual void decaf::util::logging::SimpleLogger::mark ( const std::string & message )` [virtual]

Log a Mark Block **Level** (p. 2290) Log.

6.745.2.7 `virtual void decaf::util::logging::SimpleLogger::warn ( const std::string & file, const int line, const std::string & message )` [virtual]

Log a Warning **Level** (p. 2290) Log.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleLogger.h`

## 6.746 decaf::net::Socket Class Reference

```
#include <src/main/decaf/net/Socket.h>
```

Inheritance diagram for decaf::net::Socket:

## Public Member Functions

- **Socket** ()  
*Creates an unconnected **Socket** (p. 3445) using the set **SocketImplFactory** (p. 3481) or if non is set than the default **SocketImpl** type is created.*
- **Socket** (**SocketImpl** \*impl)  
*Creates a **Socket** (p. 3445) wrapping the provided **SocketImpl** (p. 3472) instance, this **Socket** (p. 3445) is considered unconnected.*
- **Socket** (const **InetAddress** \*address, int port)  
*Creates a new **Socket** (p. 3445) instance and connects it to the given address and port.*
- **Socket** (const **InetAddress** \*address, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **Socket** (p. 3445) instance and connects it to the given address and port.*
- **Socket** (const std::string &host, int port)  
*Creates a new **Socket** (p. 3445) instance and connects it to the given host and port.*
- **Socket** (const std::string &host, int port, const **InetAddress** \*localAddress, int localPort)  
*Creates a new **Socket** (p. 3445) instance and connects it to the given host and port.*
- virtual ~**Socket** ()
- virtual void **bind** (const std::string &ipaddress, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Binds this **Socket** (p. 3445) to the given local address and port.*
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes the **Socket** (p. 3445).*
- virtual void **connect** (const std::string &host, int port) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects to the specified destination.*
- virtual void **connect** (const std::string &host, int port, int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects to the specified destination, with a specified timeout value.*
- bool **isConnected** () const  
*Indicates whether or not this socket is connected to an end point.*
- bool **isClosed** () const
- bool **isBound** () const
- bool **isInputShutdown** () const
- bool **isOutputShutdown** () const
- virtual decaf::io::InputStream \* **getInputStream** () throw ( decaf::io::IOException )  
*Gets the InputStream for this socket if its connected.*

- virtual **decaf::io::OutputStream** \* **getOutputStream** () throw ( decaf::io::IOException )  
*Gets the OutputStream for this socket if it is connected.*
- int **getPort** () const  
*Gets the on the remote host this **Socket** (p. 3445) is connected to.*
- int **getLocalPort** () const  
*Gets the local port the socket is bound to.*
- std::string **getInetAddress** () const  
*Returns the address to which the socket is connected.*
- std::string **getLocalAddress** () const  
*Gets the local address to which the socket is bound.*
- virtual void **shutdownInput** () throw ( decaf::io::IOException )  
*Shuts down the InputStream for this socket essentially marking it as EOF.*
- virtual void **shutdownOutput** () throw ( decaf::io::IOException )  
*Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to OuputStream::write will throw an IOException.*
- virtual int **getSoLinger** () const throw ( SocketException )  
*Gets the linger time for the socket, SO\_LINGER.*
- virtual void **setSoLinger** (bool state, int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the linger time (SO\_LINGER) using a specified time value, this limits of this value are platform specific.*
- virtual bool **getKeepAlive** () const throw ( SocketException )  
*Gets the keep alive flag for this socket, SO\_KEEPALIVE.*
- virtual void **setKeepAlive** (bool keepAlive) throw ( SocketException )  
*Enables/disables the keep alive flag for this socket, SO\_KEEPALIVE.*
- virtual int **getReceiveBufferSize** () const throw ( SocketException )  
*Gets the receive buffer size for this socket, SO\_RCVBUF.*
- virtual void **setReceiveBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the receive buffer size for this socket, SO\_RCVBUF.*
- virtual bool **getReuseAddress** () const throw ( SocketException )  
*Gets the reuse address flag, SO\_REUSEADDR.*
- virtual void **setReuseAddress** (bool reuse) throw ( SocketException )  
*Sets the reuse address flag, SO\_REUSEADDR.*
- virtual int **getSendBufferSize** () const throw ( SocketException )  
*Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.*
- virtual void **setSendBufferSize** (int size) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.*
- virtual int **getSoTimeout** () const throw ( SocketException )  
*Gets the timeout for socket operations, SO\_TIMEOUT.*



- virtual void **setSoTimeout** (int timeout) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Sets the timeout for socket operations, SO\_TIMEOUT.*
- virtual bool **getTcpNoDelay** () const throw ( SocketException )  
*Gets the Status of the TCP\_NODELAY setting for this socket.*
- virtual void **setTcpNoDelay** (bool value) throw ( SocketException )  
*Sets the Status of the TCP\_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 3445).*
- virtual int **getTrafficClass** () const throw ( SocketException )  
*Gets the Traffic Class setting for this **Socket** (p. 3445), sometimes referred to as Type of Service setting.*
- virtual void **setTrafficClass** (int value) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )  
*Gets the Traffic Class setting for this **Socket** (p. 3445), sometimes referred to as Type of Service setting.*
- virtual bool **getOOBInline** () const throw ( SocketException )  
*Gets the value of the OOBINLINE for this socket.*
- virtual void **setOOBInline** (bool value) throw ( SocketException )  
*Sets the value of the OOBINLINE for this socket, by default this option is disabled.*
- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )  
*Sends on byte of urgent data to the **Socket** (p. 3445).*
- virtual std::string **toString** () const

### Static Public Member Functions

- static void **setSocketImplFactory** (SocketImplFactory \*factory) throw ( decaf::io::IOException, decaf::net::SocketException )  
*Sets the instance of a **SocketImplFactory** (p. 3481) that the **Socket** (p. 3445) class should use when new instances of this class are created.*

### Protected Member Functions

- void **accepted** ()
- void **initSocketImpl** (const std::string &address, int port, const InetAddress \*localAddress, int localPort) throw ( decaf::io::IOException, decaf::net::UnknownHostException )
- void **checkClosed** () const throw ( decaf::io::IOException )
- void **ensureCreated** () const throw ( decaf::io::IOException )

### Protected Attributes

- **SocketImpl \* impl**

### Friends

- class **ServerSocket**

### 6.746.1 Detailed Description

#### Since

1.0

### 6.746.2 Constructor & Destructor Documentation

#### 6.746.2.1 `decaf::net::Socket::Socket ( )`

Creates an unconnected **Socket** (p. 3445) using the set **SocketImplFactory** (p. 3481) or if non is set than the default **SocketImpl** type is created.

#### 6.746.2.2 `decaf::net::Socket::Socket ( SocketImpl * impl )`

Creates a **Socket** (p. 3445) wrapping the provided **SocketImpl** (p. 3472) instance, this **Socket** (p. 3445) is considered unconnected.

The **Socket** (p. 3445) class takes ownership of this **SocketImpl** (p. 3472) pointer and will delete it when the **Socket** (p. 3445) class is destroyed.

#### Parameters

<i>impl</i>	The <b>SocketImpl</b> (p. 3472) instance to wrap.
-------------	---

#### Exceptions

<i>NullPointerException</i>	if the passed <b>SocketImpl</b> (p. 3472) is Null.
-----------------------------	--

#### 6.746.2.3 `decaf::net::Socket::Socket ( const InetAddress * address, int port )`

Creates a new **Socket** (p. 3445) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 3481) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 3445) implementation is used.

If the host parameter is empty then the loop back address is used.

#### Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]

#### Exceptions

<b>UnknownHostException</b> (p. 3841)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).
<i>NullPointerException</i>	if the <b>InetAddress</b> (p. 1974) instance is NULL.

<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]
---------------------------------------	---

**6.746.2.4** decaf::net::Socket::Socket ( const InetAddress \* address, int port, const InetAddress \* localAddress, int localPort )

Creates a new **Socket** (p. 3445) instance and connects it to the given address and port.

If there is a **SocketImplFactory** (p. 3481) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 3445) implementation is used. The **Socket** (p. 3445) will also bind to the local address and port specified.

#### Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

#### Exceptions

<b>UnknownHostException</b> (p. 3841)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).
<i>NullPointerException</i>	if the <b>InetAddress</b> (p. 1974) instance is NULL.
<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]

**6.746.2.5** decaf::net::Socket::Socket ( const std::string & host, int port )

Creates a new **Socket** (p. 3445) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 3481) set then the SokcetImpl is created using the factory otherwise the default **Socket** (p. 3445) implementation is used.

If the host parameter is empty then the loop back address is used.

#### Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]

#### Exceptions

<b>UnknownHostException</b> (p. 3841)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).

<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]
---------------------------------------	---

6.746.2.6 `decaf::net::Socket::Socket ( const std::string & host, int port, const InetAddress * localAddress, int localPort )`

Creates a new **Socket** (p. 3445) instance and connects it to the given host and port.

If there is a **SocketImplFactory** (p. 3481) set then the **SocketImpl** is created using the factory otherwise the default **Socket** (p. 3445) implementation is used.

If the host parameter is empty then the loop back address is used.

#### Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

#### Exceptions

<b>UnknownHostException</b> (p. 3841)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).
<i>IllegalArgumentEx- ception</i>	if the port if not in range [0...65535]

6.746.2.7 `virtual decaf::net::Socket::~~Socket ( ) [virtual]`

### 6.746.3 Member Function Documentation

6.746.3.1 `void decaf::net::Socket::accepted ( ) [protected]`

6.746.3.2 `virtual void decaf::net::Socket::bind ( const std::string & ipaddress, int port ) throw ( decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Binds this **Socket** (p. 3445) to the given local address and port.

If the **SocketAddress** (p. 3463) value is NULL then the **Socket** (p. 3445) will be bound to an available local address and port.

#### Parameters

<i>ipaddress</i>	The local address and port to bind the socket to.
<i>port</i>	The port on the local machine to bind to.

**Exceptions**

<i>IOException</i>	if an error occurs during the bind operation.
<i>IllegalArgumentEx- ception</i>	if the <b>Socket</b> (p. 3445) can't process the subclass of <b>SocketAd- dress</b> (p. 3463) that has been provided.

**6.746.3.3** void decaf::net::Socket::checkClosed ( ) const throw ( decaf::io::IOException )  
[protected]

**6.746.3.4** virtual void decaf::net::Socket::close ( ) throw ( decaf::io::IOException )  
[virtual]

Closes the **Socket** (p. 3445).

Once closed a **Socket** (p. 3445) cannot be connected or otherwise operated upon, a new **Socket** (p. 3445) instance must be created.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while closing the <b>Socket</b> (p. 3445).
--------------------	---

Implements **decaf::io::Closeable** (p. 1121).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2813).

**6.746.3.5** virtual void decaf::net::Socket::connect ( const std::string &  
*host*, int *port*, int *timeout* ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Connects to the specified destination, with a specified timeout value.

If a connection to the remote host is not established within the specified timeout interval than an **SocketTimeoutException** (p. 3487) is thrown. A timeout value of zero is treated as an infinite timeout.

**Parameters**

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.
<i>timeout</i>	The number of Milliseconds to wait before treating the connection as failed.

**Exceptions**

<i>IOException</i>	Thrown if a failure occurred in the connect.
<b>SocketTimeoutEx- ception</b> (p. 3487)	if the timeout for connection is exceeded.
<i>IllegalArguementEx- ception</i>	if the timeout value is negative or the endpoint is invalid.

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2813).

6.746.3.6 `virtual void decaf::net::Socket::connect ( const std::string  
& host, int port ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::IllegalArgumentException ) [virtual]`

Connects to the specified destination.

#### Parameters

<i>host</i>	The host name or IP address of the remote host to connect to.
<i>port</i>	The port on the remote host to connect to.

#### Exceptions

<i>IOException</i>	Thrown if a failure occurred in the connect.
<i>IllegalArgumentEx- ception</i>	if the timeout value is negative or the endpoint is invalid.

6.746.3.7 `void decaf::net::Socket::ensureCreated ( ) const throw ( decaf::io::IOException )  
[protected]`

6.746.3.8 `std::string decaf::net::Socket::getInetAddress ( ) const`

Returns the address to which the socket is connected.

#### Returns

the remote IP address to which this socket is connected, or null if the socket is not connected.

6.746.3.9 `virtual decaf::io::InputStream* decaf::net::Socket::getInputStream ( ) throw ( decaf::io::IOException ) [virtual]`

Gets the InputStream for this socket if its connected.

The pointer returned is the property of the associated **Socket** (p. 3445) and should not be deleted by the caller.

When the returned InputStream is performing a blocking operation and the underlying connection is closed or otherwise broken the read calls will normally throw an exception to indicate the failure.

Closing the InputStream will also close the underlying **Socket** (p. 3445).

#### Returns

The InputStream for this socket.

**Exceptions**

<i>IOException</i>	if an error occurs during creation of the InputStream, also if the <b>Socket</b> (p. 3445) is not connected or the input has been shutdown previously.
--------------------	--

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2814).

**6.746.3.10** `virtual bool decaf::net::Socket::getKeepAlive ( ) const throw ( SocketException )`  
[virtual]

Gets the keep alive flag for this socket, SO\_KEEPALIVE.

**Returns**

true if keep alive is enabled for this socket.

**Exceptions**

<b>SocketException</b> (p. 3465)	if the operation fails.
-------------------------------------	-------------------------

**6.746.3.11** `std::string decaf::net::Socket::getLocalAddress ( ) const`

Gets the local address to which the socket is bound.

**Returns**

the local address to which the socket is bound or InetAddress.anyLocalAddress() if the socket is not bound yet.

**6.746.3.12** `int decaf::net::Socket::getLocalPort ( ) const`

Gets the local port the socket is bound to.

**Returns**

the local port the socket was bound to, or -1 if the socket is not bound.

**6.746.3.13** `virtual bool decaf::net::Socket::getOOBInline ( ) const throw ( SocketException )`  
[virtual]

Gets the value of the OOBINLINE for this socket.

**Returns**

true if OOBINLINE is enabled, false otherwise.

**Exceptions**

<b><i>SocketException</i></b> (p. 3465)	if an error is encountered while performing this operation.
--	---

6.746.3.14 `virtual decaf::io::OutputStream* decaf::net::Socket::getOutputStream ( ) throw ( decaf::io::IOException ) [virtual]`

Gets the OutputStream for this socket if it is connected.

The pointer returned is the property of the **Socket** (p. 3445) instance and should not be deleted by the caller.

Closing the returned **Socket** (p. 3445) will also close the underlying **Socket** (p. 3445).

**Returns**

the OutputStream for this socket.

**Exceptions**

<i>IOException</i>	if an error occurs during the creation of this OutputStream, or if the <b>Socket</b> (p. 3445) is closed or the output has been shutdown previously.
--------------------	--

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2815).

6.746.3.15 `int decaf::net::Socket::getPort ( ) const`

Gets the on the remote host this **Socket** (p. 3445) is connected to.

**Returns**

the port on the remote host the socket is connected to, or 0 if not connected.

6.746.3.16 `virtual int decaf::net::Socket::getReceiveBufferSize ( ) const throw ( SocketException ) [virtual]`

Gets the receive buffer size for this socket, SO\_RCVBUF.

This is the buffer used by the underlying platform socket to buffer received data.

**Returns**

the receive buffer size in bytes.

**Exceptions**

<b><i>SocketException</i></b> (p. 3465)	if the operation fails.
--	-------------------------



6.746.3.17 virtual bool decaf::net::Socket::getReuseAddress ( ) const throw ( **SocketException** ) [virtual]

Gets the reuse address flag, SO\_REUSEADDR.

#### Returns

True if the address can be reused.

#### Exceptions

<b>SocketException</b> (p. 3465)	if the operation fails.
-------------------------------------	-------------------------

6.746.3.18 virtual int decaf::net::Socket::getSendBufferSize ( ) const throw ( **SocketException** ) [virtual]

Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

#### Returns

the size in bytes of the send buffer.

#### Exceptions

<b>SocketException</b> (p. 3465)	if the operation fails.
-------------------------------------	-------------------------

6.746.3.19 virtual int decaf::net::Socket::getSoLinger ( ) const throw ( **SocketException** ) [virtual]

Gets the linger time for the socket, SO\_LINGER.

A return value of -1 indicates that the option is disabled.

#### Returns

The linger time in seconds.

#### Exceptions

<b>SocketException</b> (p. 3465)	if the operation fails.
-------------------------------------	-------------------------

6.746.3.20 `virtual int decaf::net::Socket::getSoTimeout ( ) const throw ( SocketException )`  
`[virtual]`

Gets the timeout for socket operations, SO\_TIMEOUT.

#### Returns

The timeout in milliseconds for socket operations.

#### Exceptions

<b>SocketException</b> (p. 3465)	Thrown if unable to retrieve the information.
-------------------------------------	---

6.746.3.21 `virtual bool decaf::net::Socket::getTcpNoDelay ( ) const throw ( SocketException )`  
`[virtual]`

Gets the Status of the TCP\_NODELAY setting for this socket.

#### Returns

true if TCP\_NODELAY is enabled for the socket.

#### Exceptions

<b>SocketException</b> (p. 3465)	Thrown if unable to set the information.
-------------------------------------	--

6.746.3.22 `virtual int decaf::net::Socket::getTrafficClass ( ) const throw ( SocketException )`  
`[virtual]`

Gets the Traffic Class setting for this **Socket** (p. 3445), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3445) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

#### Returns

the bitset result of querying the traffic class setting.

#### Exceptions

<b>SocketException</b> (p. 3465)	if an error is encountered while performing this operation.
-------------------------------------	---

6.746.3.23 void decaf::net::Socket::initSocketImpl ( const std::string & *address*, int *port*, const InetAddress \* *localAddress*, int *localPort* ) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [protected]

6.746.3.24 bool decaf::net::Socket::isBound ( ) const [inline]

#### Returns

true if this **Socket** (p. 3445) has been bound to a Local address.

6.746.3.25 bool decaf::net::Socket::isClosed ( ) const [inline]

#### Returns

true if the **Socket** (p. 3445) has been closed.

6.746.3.26 bool decaf::net::Socket::isConnected ( ) const [inline]

Indicates whether or not this socket is connected to an end point.

#### Returns

true if connected, false otherwise.

6.746.3.27 bool decaf::net::Socket::isInputShutdown ( ) const [inline]

#### Returns

true if input on this **Socket** (p. 3445) has been shutdown.

6.746.3.28 bool decaf::net::Socket::isOutputShutdown ( ) const [inline]

#### Returns

true if output on this **Socket** (p. 3445) has been shutdown.

6.746.3.29 virtual void decaf::net::Socket::sendUrgentData ( int *data* ) throw ( decaf::io::IOException ) [virtual]

Sends one byte of urgent data to the **Socket** (p. 3445).

#### Parameters

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2817).

6.746.3.30 **virtual void decaf::net::Socket::setKeepAlive ( bool *keepAlive* ) throw ( **SocketException** )** [virtual]

Enables/disables the keep alive flag for this socket, SO\_KEEPALIVE.

**Parameters**

<i>keepAlive</i>	If true, enables the flag.
------------------	----------------------------

**Exceptions**

<b>SocketException</b> (p. 3465)	if the operation fails.
-------------------------------------	-------------------------

6.746.3.31 **virtual void decaf::net::Socket::setOOBInline ( bool *value* ) throw ( **SocketException** )** [virtual]

Sets the value of the OOBINLINE for this socket, by default this option is disabled.

If enabled the urgent data is read inline on the Socket's InputStream, no notification is give.

**Returns**

true if OOBINLINE is enabled, false otherwise.

**Exceptions**

<b>SocketException</b> (p. 3465)	if an error is encountered while performing this operation.
-------------------------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2819).

6.746.3.32 **virtual void decaf::net::Socket::setReceiveBufferSize ( int *size* ) throw ( **SocketException**, **decaf::lang::exceptions::IllegalArgumentException** )** [virtual]

Sets the receive buffer size for this socket, SO\_RCVBUF.

**Parameters**

<i>size</i>	Number of bytes to set the receive buffer to.
-------------	---

**Exceptions**

<b><i>SocketException</i></b> (p. 3465)	if the operation fails.
<i>IllegalArgumentEx- ception</i>	if the value is zero or negative.

6.746.3.33 virtual void decaf::net::Socket::setReuseAddress ( bool *reuse* ) throw ( **SocketException** ) [virtual]

Sets the reuse address flag, SO\_REUSEADDR.

**Parameters**

<i>reuse</i>	If true, sets the flag.
--------------	-------------------------

**Exceptions**

<b><i>SocketException</i></b> (p. 3465)	if the operation fails.
--	-------------------------

6.746.3.34 virtual void decaf::net::Socket::setSendBufferSize ( int *size* ) throw ( **SocketException**, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Gets the send buffer size for this socket, SO\_SNDBUF, this value is used by the platform socket to buffer data written to the socket.

**Parameters**

<i>size</i>	The number of bytes to set the send buffer to, must be larger than zero.
-------------	--

**Exceptions**

<b><i>SocketException</i></b> (p. 3465)	if the operation fails.
<i>IllegalArgumentEx- ception</i>	if the value is zero or negative.

6.746.3.35 static void decaf::net::Socket::setSocketImplFactory ( **SocketImplFactory** \* *factory* ) throw ( decaf::io::IOException, decaf::net::SocketException ) [static]

Sets the instance of a **SocketImplFactory** (p.3481) that the **Socket** (p.3445) class should use when new instances of this class are created.

This method is only allowed to be used once during the lifetime of the application.

**Parameters**

<i>factory</i>	The instance of a <b>SocketImplFactory</b> (p. 3481) to use when new <b>Socket</b> (p. 3445) objects are created.
----------------	---

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
<b>SocketException</b> (p. 3465)	if this method has already been called with a valid factory.

6.746.3.36 `virtual void decaf::net::Socket::setSoLinger ( bool state, int timeout ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
[virtual]

Sets the linger time (SO\_LINGER) using a specified time value, this limits of this value are platform specific.

**Parameters**

<i>state</i>	The state of SO_LINGER, true is on.
<i>timeout</i>	The linger time in seconds, must be non-negative.

**Exceptions**

<b>SocketException</b> (p. 3465)	if the operation fails.
<i>IllegalArgumentException</i>	if state is true and timeout is negative.

6.746.3.37 `virtual void decaf::net::Socket::setSoTimeout ( int timeout ) throw ( SocketException, decaf::lang::exceptions::IllegalArgumentException )`  
[virtual]

Sets the timeout for socket operations, SO\_TIMEOUT.

A value of zero indicates that timeout is infinite for operations on this socket.

**Parameters**

<i>timeout</i>	The timeout in milliseconds for socket operations.
----------------	--

**Exceptions**

<b>SocketException</b> (p. 3465)	Thrown if unable to set the information.
<i>IllegalArgumentException</i>	if the timeout value is negative.

6.746.3.38 virtual void decaf::net::Socket::setTcpNoDelay ( bool *value* ) throw ( **SocketException** ) [virtual]

Sets the Status of the TCP\_NODELAY param for this socket., this setting is used to disable or enable Nagle's algorithm on the **Socket** (p. 3445).

#### Parameters

<i>value</i>	The setting for the socket's TCP_NODELAY option, true to enable.
--------------	--

#### Exceptions

<b>SocketException</b> (p. 3465)	Thrown if unable to set the information.
-------------------------------------	--

6.746.3.39 virtual void decaf::net::Socket::setTrafficClass ( int *value* ) throw ( **SocketException**, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Gets the Traffic Class setting for this **Socket** (p. 3445), sometimes referred to as Type of Service setting.

This setting is dependent on the underlying network implementation for the platform this **Socket** (p. 3445) runs on and is not guaranteed to have any effect.

Refer to your platforms network documentation regarding support for this setting.

#### Parameters

<i>value</i>	The integer value representing the traffic class setting bitset.
--------------	--

#### Exceptions

<b>SocketException</b> (p. 3465)	if an error is encountered while performing this operation.
<i>IllegalArgumentException</i>	if the value is not in the range [0..255].

6.746.3.40 virtual void decaf::net::Socket::shutdownInput ( ) throw ( decaf::io::IOException ) [virtual]

Shuts down the InputStream for this socket essentially marking it as EOF.

The stream returns EOF for any calls to read after this method has been called.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2820).

6.746.3.41 `virtual void decaf::net::Socket::shutdownOutput ( ) throw ( decaf::io::IOException )` `[virtual]`

Shuts down the OutputStream for this socket, any data already written to the socket will be sent, any further calls to `OuputStream::write` will throw an `IOException`.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Reimplemented in `decaf::internal::net::ssl::openssl::OpenSSLSocket` (p. 2820).

6.746.3.42 `virtual std::string decaf::net::Socket::toString ( ) const` `[virtual]`

#### Returns

a string representing this **Socket** (p. 3445).

### 6.746.4 Friends And Related Function Documentation

6.746.4.1 `friend class ServerSocket` `[friend]`

### 6.746.5 Field Documentation

6.746.5.1 `SocketImpl* decaf::net::Socket::impl` `[mutable, protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Socket.h`

## 6.747 decaf::net::SocketAddress Class Reference

Base class for protocol specific **Socket** (p. 3445) addresses.

```
#include <src/main/decaf/net/SocketAddress.h>
```

Inheritance diagram for `decaf::net::SocketAddress`:

#### Public Member Functions

- `virtual ~SocketAddress ( )`



### 6.747.1 Detailed Description

Base class for protocol specific **Socket** (p. 3445) addresses.

These classes provide an immutable address object that is used by the **Socket** (p. 3445) classes.

#### Since

1.0

### 6.747.2 Constructor & Destructor Documentation

6.747.2.1 `virtual decaf::net::SocketAddress::~~SocketAddress ( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketAddress.h`

## 6.748 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

```
#include <src/main/decaf/net/SocketError.h>
```

### Static Public Member Functions

- static int **getErrorCode** ()  
*Gets the last error appropriate for the platform.*
- static std::string **getErrorString** ()  
*Gets the string description for the last error.*

### 6.748.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

### 6.748.2 Member Function Documentation

6.748.2.1 `static int decaf::net::SocketError::getErrorCode ( ) [static]`

Gets the last error appropriate for the platform.

6.748.2.2 `static std::string decaf::net::SocketError::getErrorString ( ) [static]`

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketError.h`

## 6.749 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

```
#include <src/main/decaf/net/SocketException.h>
```

Inheritance diagram for `decaf::net::SocketException`:

### Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const **lang::Exception** &ex) throw ()
- **SocketException** (const **SocketException** &ex) throw ()
- **SocketException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SocketException** (const std::exception \*cause) throw ()  
*Constructor.*
- **SocketException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SocketException** \* clone () const  
*Clones this exception.*
- virtual ~**SocketException** () throw ()

### 6.749.1 Detailed Description

Exception for errors when manipulating sockets.

### 6.749.2 Constructor & Destructor Documentation

6.749.2.1 `decaf::net::SocketException::SocketException ( ) throw () [inline]`

6.749.2.2 `decaf::net::SocketException::SocketException ( const lang::Exception & ex ) throw () [inline]`

**6.749.2.3** `decaf::net::SocketException::SocketException ( const SocketException & ex )  
throw () [inline]`

**6.749.2.4** `decaf::net::SocketException::SocketException ( const char * file, const int  
lineNumber, const std::exception * cause, const char * msg, ... ) throw ()  
[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.749.2.5** `decaf::net::SocketException::SocketException ( const std::exception * cause ) throw  
() [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.749.2.6** `decaf::net::SocketException::SocketException ( const char * file, const int  
lineNumber, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

**6.749.2.7** `virtual decaf::net::SocketException::~~SocketException ( ) throw () [inline,  
virtual]`

### 6.749.3 Member Function Documentation

6.749.3.1 **virtual SocketException\*** **decaf::net::SocketException::clone** ( ) const  
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2105).

Reimplemented in **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2824), **decaf::net::BindException** (p. 800), **decaf::net::ConnectException** (p. 1232), **decaf::net::NoRouteToHostException** (p. 2775), and **decaf::net::PortUnreachableException** (p. 2924).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketException.h**

### 6.750 decaf::net::SocketFactory Class Reference

The **SocketFactory** (p. 3467) is used to create **Socket** (p. 3445) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

```
#include <src/main/decaf/net/SocketFactory.h>
```

Inheritance diagram for decaf::net::SocketFactory:

#### Public Member Functions

- virtual **~SocketFactory** ()
- virtual **Socket \* createSocket** () throw ( decaf::io::IOException )  
*Creates an unconnected **Socket** (p. 3445) object.*
- virtual **Socket \* createSocket** (const **InetAddress** \*host, int port)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )  
*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*
- virtual **Socket \* createSocket** (const **InetAddress** \*host, int port, const **InetAddress \*ifAddress**, int localPort)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

- virtual **Socket** \* **createSocket** (const std::string &name, int port)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

- virtual **Socket** \* **createSocket** (const std::string &name, int port, const **InetAddress** \*ifAddress, int localPort)=0 throw ( decaf::io::IOException, decaf::net::UnknownHostException )

*Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).*

## Static Public Member Functions

- static **SocketFactory** \* **getDefault** ()

*Returns an pointer to the default **SocketFactory** (p. 3467) for this Application, there is only one default **SocketFactory** (p. 3467) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3467) class and in not to be deleted by the caller.*

## Protected Member Functions

- **SocketFactory** ()

### 6.750.1 Detailed Description

The **SocketFactory** (p. 3467) is used to create **Socket** (p. 3445) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.

#### See also

**decaf.net.Socket** (p. 3445)

#### Since

1.0

### 6.750.2 Constructor & Destructor Documentation

6.750.2.1 decaf::net::SocketFactory::SocketFactory ( ) [protected]

6.750.2.2 virtual decaf::net::SocketFactory::~~SocketFactory ( ) [virtual]

### 6.750.3 Member Function Documentation

6.750.3.1 `virtual Socket* decaf::net::SocketFactory::createSocket ( ) throw ( decaf::io::IOException ) [virtual]`

Creates an unconnected **Socket** (p. 3445) object.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if the <b>Socket</b> (p. 3445) cannot be created.
--------------------	---

Reimplemented in **decaf::internal::net::DefaultSocketFactory** (p. 1654), **decaf::internal::net::ssl::DefaultSSLFactory** (p. 1666), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2828).

6.750.3.2 `virtual Socket* decaf::net::SocketFactory::createSocket ( const InetAddress * host, int port ) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [pure virtual]`

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1656), **decaf::internal::net::ssl::DefaultSSLFactory** (p. 1667), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2829).

6.750.3.3 `virtual Socket* decaf::net::SocketFactory::createSocket ( const std::string & name, int port, const InetAddress * ifAddress, int localPort ) throw ( decaf::io::IOException, decaf::net::UnknownHostException ) [pure virtual]`

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

**Parameters**

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1654), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1668), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2830).

```
6.750.3.4 virtual Socket* decaf::net::SocketFactory::createSocket ( const
std::string & name, int port ) throw ( decaf::io::IOException,
decaf::net::UnknownHostException ) [pure virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

**Parameters**

<i>host</i>	The host name or IP address to connect the socket to.
<i>port</i>	The port on the remote host to connect to.

**Returns**

a new **Socket** (p. 3445) object, caller must free this object when done.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1655), **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1668), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2830).

```
6.750.3.5 virtual Socket* decaf::net::SocketFactory::createSocket ( const InetAddress
* host, int port, const InetAddress * ifAddress, int localPort ) throw (
decaf::io::IOException, decaf::net::UnknownHostException ) [pure
virtual]
```

Creates a new **Socket** (p. 3445) object and connects it to the specified remote host and port using the configuration of this **SocketFactory** (p. 3467).

The **Socket** (p. 3445) will be bound to the specified local address and port.

#### Parameters

<i>host</i>	The host to connect the socket to.
<i>port</i>	The port on the remote host to connect to.
<i>ifAddress</i>	The address on the local machine to bind the <b>Socket</b> (p. 3445) to.
<i>localPort</i>	The local port to bind the <b>Socket</b> (p. 3445) to.

#### Returns

a new **Socket** (p. 3445) object, caller must free this object when done.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while creating the <b>Socket</b> (p. 3445) object.
<b>UnknownHostException</b> (p. 3841)	if the host name is not known.

Implemented in **decaf::internal::net::DefaultSocketFactory** (p. 1656), **decaf::internal::net::ssl::DefaultSSLFactory** (p. 1669), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2831).

```
6.750.3.6 static SocketFactory* decaf::net::SocketFactory::getDefault ( ) [static]
```

Returns a pointer to the default **SocketFactory** (p. 3467) for this Application, there is only one default **SocketFactory** (p. 3467) per application, the pointer returned by this method is owned by the **SocketFactory** (p. 3467) class and in not to be deleted by the caller.

#### Returns

pointer to the applications default **SocketFactory** (p. 3467).

#### Exceptions

<b>SocketException</b> (p. 3465)	if an error occurs while getting the default instance.
-------------------------------------	--

Reimplemented in **decaf::net::ssl::SSLSocketFactory** (p. 3517).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketFactory.h**



## 6.751 decaf::internal::net::SocketFileDescriptor Class Reference

File Descriptor type used internally by Decaf Socket objects.

```
#include <src/main/decaf/internal/net/SocketFileDescriptor.h>
```

Inheritance diagram for decaf::internal::net::SocketFileDescriptor:

### Public Member Functions

- **SocketFileDescriptor** (long value)
- virtual **~SocketFileDescriptor** ()
- long **getValue** () const

*Gets the OS Level FileDescriptor.*

#### 6.751.1 Detailed Description

File Descriptor type used internally by Decaf Socket objects.

#### Since

1.0

#### 6.751.2 Constructor & Destructor Documentation

6.751.2.1 decaf::internal::net::SocketFileDescriptor::SocketFileDescriptor ( long value )

6.751.2.2 virtual decaf::internal::net::SocketFileDescriptor::~~SocketFileDescriptor ( )  
[virtual]

#### 6.751.3 Member Function Documentation

6.751.3.1 long decaf::internal::net::SocketFileDescriptor::getValue ( ) const

Gets the OS Level FileDescriptor.

#### Returns

a FileDescriptor value.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**SocketFileDescriptor.h**

## 6.752 decaf::net::SocketImpl Class Reference

Acts as a base class for all physical **Socket** (p. 3445) implementations.

```
#include <src/main/decaf/net/SocketImpl.h>
```

Inheritance diagram for decaf::net::SocketImpl:

### Public Member Functions

- **SocketImpl** ()
- virtual **~SocketImpl** ()
- virtual void **create** ()=0 throw ( decaf::io::IOException )  
*Creates the underlying platform **Socket** (p. 3445) data structures which allows for **Socket** (p. 3445) options to be applied.*
- virtual void **accept** (**SocketImpl** \*socket)=0 throw ( decaf::io::IOException, decaf::net::SocketException, decaf::net::SocketTimeoutException )  
*Accepts a new connection on the given **Socket** (p. 3445).*
- virtual void **connect** (const std::string &hostname, int **port**, int timeout)=0 throw ( decaf::io::IOException, decaf::net::SocketTimeoutException, decaf::lang::exceptions::IllegalArgumentException )  
*Connects this socket to the given host and port.*
- virtual void **bind** (const std::string &ipaddress, int **port**)=0 throw ( decaf::io::IOException )  
*Binds this **Socket** (p. 3445) instance to the local ip address and port number given.*
- virtual void **listen** (int backlog)=0 throw ( decaf::io::IOException )  
*Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.*
- virtual decaf::io::InputStream \* **getInputStream** ()=0 throw ( decaf::io::IOException )  
*Gets the InputStream linked to this **Socket** (p. 3445).*
- virtual decaf::io::OutputStream \* **getOutputStream** ()=0 throw ( decaf::io::IOException )  
*Gets the OutputStream linked to this **Socket** (p. 3445).*
- virtual int **available** ()=0 throw ( decaf::io::IOException )  
*Gets the number of bytes that can be read from the **Socket** (p. 3445) without blocking.*
- virtual void **close** ()=0 throw ( decaf::io::IOException )  
*Closes the socket, terminating any blocked reads or writes.*
- virtual void **shutdownInput** ()=0 throw ( decaf::io::IOException )  
*Places the input stream for this socket at "end of stream".*
- virtual void **shutdownOutput** ()=0 throw ( decaf::io::IOException )  
*Disables the output stream for this socket.*
- virtual int **getOption** (int option) const =0 throw ( decaf::io::IOException )  
*Gets the specified **Socket** (p. 3445) option.*

- virtual void **setOption** (int option, int value)=0 throw ( decaf::io::IOException )  
*Sets the specified option on the **Socket** (p. 3445) if supported.*
- int **getPort** () const  
*Gets the port that this socket has been assigned.*
- int **getLocalPort** () const  
*Gets the value of this SocketImpl's local port field.*
- std::string **getInetAddress** () const  
*Gets the value of this SocketImpl's address field.*
- const decaf::io::FileDescriptor \* **getFileDescriptor** () const  
*Gets the FileDescriptor for this **Socket** (p. 3445), the Object is owned by this **Socket** (p. 3445) and should not be deleted by the caller.*
- virtual std::string **getLocalAddress** () const =0  
*Gets the value of the local Inet address the **Socket** (p. 3445) is bound to if bound, otherwise return the **InetAddress** (p. 1974) ANY value "0.0.0.0".*
- std::string **toString** () const  
*Returns a string containing the address and port of this **Socket** (p. 3445) instance.*
- virtual bool **supportsUrgentData** () const
- virtual void **sendUrgentData** (int data) throw ( decaf::io::IOException )  
*Sends on byte of urgent data to the **Socket** (p. 3445).*

### Protected Attributes

- int **port**  
*The remote port that this **Socket** (p. 3445) is connected to.*
- int **localPort**  
*The port on the Local Machine that this **Socket** (p. 3445) is Bound to.*
- std::string **address**  
*The Remote Address that the **Socket** (p. 3445) is connected to.*
- io::FileDescriptor \* **fd**  
*The File Descriptor for this **Socket** (p. 3445).*

### 6.752.1 Detailed Description

Acts as a base class for all physical **Socket** (p. 3445) implementations.

Since

1.0

## 6.752.2 Constructor & Destructor Documentation

6.752.2.1 `decaf::net::SocketImpl::SocketImpl ( )`

6.752.2.2 `virtual decaf::net::SocketImpl::~~SocketImpl ( )` `[virtual]`

## 6.752.3 Member Function Documentation

6.752.3.1 `virtual void decaf::net::SocketImpl::accept ( SocketImpl * socket )  
throw ( decaf::io::IOException, decaf::net::SocketException,  
decaf::net::SocketTimeoutException )` `[pure virtual]`

Accepts a new connection on the given **Socket** (p. 3445).

### Parameters

<i>socket</i>	The accepted connection.
---------------	--------------------------

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
<b>SocketException</b> (p. 3465)	if an error occurs while performing an Accept on the socket.
<b>SocketTimeoutException</b> (p. 3487)	if the accept call times out due to SO_TIMEOUT being set.

6.752.3.2 `virtual int decaf::net::SocketImpl::available ( )` throw ( `decaf::io::IOException` )  
`[pure virtual]`

Gets the number of bytes that can be read from the **Socket** (p. 3445) without blocking.

### Returns

the number of bytes that can be read from the **Socket** (p. 3445) without blocking.

### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3685).

6.752.3.3 `virtual void decaf::net::SocketImpl::bind ( const std::string & ipaddress, int port )  
throw ( decaf::io::IOException )` `[pure virtual]`

Binds this **Socket** (p. 3445) instance to the local ip address and port number given.

**Parameters**

<i>ipaddress</i>	The address of local ip to bind to.
<i>port</i>	The port number on the host to bind to.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3685).

6.752.3.4 `virtual void decaf::net::SocketImpl::close ( ) throw ( decaf::io::IOException )`  
`[pure virtual]`

Closes the socket, terminating any blocked reads or writes.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3686).

6.752.3.5 `virtual void decaf::net::SocketImpl::connect ( const`  
`std::string & hostname, int port, int timeout ) throw (`  
`decaf::io::IOException, decaf::net::SocketTimeoutException,`  
`decaf::lang::exceptions::IllegalArgumentException ) [pure`  
`virtual]`

Connects this socket to the given host and port.

**Parameters**

<i>hostname</i>	The name of the host to connect to, or IP address.
<i>port</i>	The port number on the host to connect to.
<i>timeout</i>	Time in milliseconds to wait for a connection, 0 indicates forever.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
<b>SocketTimeoutException</b> (p. 3487)	if the connect call times out due to timeout being set.
<i>IllegalArgumentEx-ception</i>	if a parameter has an illegal value.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3686).

6.752.3.6 `virtual void decaf::net::SocketImpl::create ( ) throw ( decaf::io::IOException )`  
`[pure virtual]`

Creates the underlying platform **Socket** (p. 3445) data structures which allows for **Socket** (p. 3445) options to be applied.

The created socket is in an unconnected state.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3686).

6.752.3.7 `const decaf::io::FileDescriptor* decaf::net::SocketImpl::getFileDescriptor ( )`  
`const [inline]`

Gets the FileDescriptor for this **Socket** (p. 3445), the Object is owned by this **Socket** (p. 3445) and should not be deleted by the caller.

#### Returns

a pointer to this Socket's FileDescriptor object.

6.752.3.8 `std::string decaf::net::SocketImpl::getInetAddress ( ) const [inline]`

Gets the value of this SocketImpl's address field.

#### Returns

the value of the address field.

6.752.3.9 `virtual decaf::io::InputStream* decaf::net::SocketImpl::getInputStream ( ) throw`  
`( decaf::io::IOException ) [pure virtual]`

Gets the InputStream linked to this **Socket** (p. 3445).

#### Returns

an InputStream pointer owned by the **Socket** (p. 3445) object.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3687).

6.752.3.10 `virtual std::string decaf::net::SocketImpl::getLocalAddress ( ) const [pure virtual]`

Gets the value of the local Inet address the **Socket** (p. 3445) is bound to if bound, otherwise return the **InetAddress** (p. 1974) ANY value "0.0.0.0".

#### Returns

the local address bound to, or ANY.

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3687).

6.752.3.11 `int decaf::net::SocketImpl::getLocalPort ( ) const [inline]`

Gets the value of this SocketImpl's local port field.

#### Returns

the value of localPort.

6.752.3.12 `virtual int decaf::net::SocketImpl::getOption ( int option ) const throw ( decaf::io::IOException ) [pure virtual]`

Gets the specified **Socket** (p. 3445) option.

#### Parameters

<i>option</i>	The <b>Socket</b> (p. 3445) options whose value is to be retrieved.
---------------	---

#### Returns

the value of the given socket option.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3687).

6.752.3.13 `virtual decaf::io::OutputStream* decaf::net::SocketImpl::getOutputStream ( ) throw ( decaf::io::IOException ) [pure virtual]`

Gets the OutputStream linked to this **Socket** (p. 3445).

#### Returns

an OutputStream pointer owned by the **Socket** (p. 3445) object.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3688).

6.752.3.14 `int decaf::net::SocketImpl::getPort ( ) const [inline]`

Gets the port that this socket has been assigned.

**Returns**

the Socket's port number.

6.752.3.15 `virtual void decaf::net::SocketImpl::listen ( int backlog ) throw ( decaf::io::IOException ) [pure virtual]`

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

**Parameters**

<i>backlog</i>	The maximum length of the connection queue.
----------------	---

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3688).

6.752.3.16 `virtual void decaf::net::SocketImpl::sendUrgentData ( int data ) throw ( decaf::io::IOException ) [virtual]`

Sends on byte of urgent data to the **Socket** (p. 3445).

**Parameters**

<i>data</i>	The value to write as urgent data, only the lower eight bits are sent.
-------------	--

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---



6.752.3.17 `virtual void decaf::net::SocketImpl::setOption ( int option, int value ) throw ( decaf::io::IOException ) [pure virtual]`

Sets the specified option on the **Socket** (p. 3445) if supported.

#### Parameters

<i>option</i>	The <b>Socket</b> (p. 3445) option to set.
<i>value</i>	The value of the socket option to apply to the socket.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3689).

6.752.3.18 `virtual void decaf::net::SocketImpl::shutdownInput ( ) throw ( decaf::io::IOException ) [pure virtual]`

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3480) on the socket, the stream will return EOF.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3690).

6.752.3.19 `virtual void decaf::net::SocketImpl::shutdownOutput ( ) throw ( decaf::io::IOException ) [pure virtual]`

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3480) on the socket, the stream will throw an IOException.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implemented in **decaf::internal::net::tcp::TcpSocket** (p. 3690).

6.752.3.20 `virtual bool decaf::net::SocketImpl::supportsUrgentData ( ) const` `[inline, virtual]`

#### Returns

true if this **SocketImpl** (p. 3472) supports sending Urgent Data. The default implementation always returns false.

6.752.3.21 `std::string decaf::net::SocketImpl::toString ( ) const`

Returns a string containing the address and port of this **Socket** (p. 3445) instance.

#### Returns

a string containing the address and port of this socket.

### 6.752.4 Field Documentation

6.752.4.1 `std::string decaf::net::SocketImpl::address` `[protected]`

The Remote Address that the **Socket** (p. 3445) is connected to.

6.752.4.2 `io::FileDescriptor* decaf::net::SocketImpl::fd` `[protected]`

The File Descriptor for this **Socket** (p. 3445).

6.752.4.3 `int decaf::net::SocketImpl::localPort` `[protected]`

The port on the Local Machine that this **Socket** (p. 3445) is Bound to.

6.752.4.4 `int decaf::net::SocketImpl::port` `[protected]`

The remote port that this **Socket** (p. 3445) is connected to.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImpl.h`

### 6.753 decaf::net::SocketImplFactory Class Reference

Factory class interface for a Factory that creates SocketImpl objects.

```
#include <src/main/decaf/net/SocketImplFactory.h>
```

## Public Member Functions

- virtual `~SocketImplFactory()`
- virtual `SocketImpl * createSocketImpl()=0`

*Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3472).*

### 6.753.1 Detailed Description

Factory class interface for a Factory that creates SocketImpl objects.

These factories can be used to create various types of Sockets, e.g. Streaming, Multicast, SSL, or platform specific variations of these types.

#### See also

**decaf::net::Socket** (p. 3445)  
**decaf::net::ServerSocket** (p. 3292)

#### Since

1.0

### 6.753.2 Constructor & Destructor Documentation

6.753.2.1 `virtual decaf::net::SocketImplFactory::~~SocketImplFactory() [inline, virtual]`

### 6.753.3 Member Function Documentation

6.753.3.1 `virtual SocketImpl* decaf::net::SocketImplFactory::createSocketImpl() [pure virtual]`

Creates a new SocketImpl instance and returns it, the caller then owns the instance and must delete it when finished with the **SocketImpl** (p. 3472).

#### Returns

new **SocketImpl** (p. 3472) instance that is owned by the caller.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketImplFactory.h`

## 6.754 decaf::net::SocketOptions Class Reference

```
#include <src/main/decaf/net/SocketOptions.h>
```

Inheritance diagram for decaf::net::SocketOptions:

## Public Member Functions

- virtual `~SocketOptions ()`

## Static Public Attributes

- static const int **SOCKET\_OPTION\_TCP\_NODELAY**  
*Disable Nagle's algorithm for this connection.*
- static const int **SOCKET\_OPTION\_BINDADDR**  
*Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).*
- static const int **SOCKET\_OPTION\_REUSEADDR**  
*Sets SO\_REUSEADDR for a socket.*
- static const int **SOCKET\_OPTION\_BROADCAST**  
*Sets SO\_BROADCAST for a socket.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_IF**  
*Set which outgoing interface on which to send multicast packets.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_IF2**  
*Same as above.*
- static const int **SOCKET\_OPTION\_IP\_MULTICAST\_LOOP**  
*This option enables or disables local loopback of multicast datagrams.*
- static const int **SOCKET\_OPTION\_IP\_TOS**  
*This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.*
- static const int **SOCKET\_OPTION\_LINGER**  
*Specify a linger-on-close timeout.*
- static const int **SOCKET\_OPTION\_TIMEOUT**  
*Set a timeout on blocking **Socket** (p. 3445) operations.*
- static const int **SOCKET\_OPTION\_SNDBUF**  
*Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.*
- static const int **SOCKET\_OPTION\_RCVBUF**  
*Set a hint the size of the underlying buffers used by the platform for incoming network I/O.*
- static const int **SOCKET\_OPTION\_KEEPAIVE**  
*When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.*
- static const int **SOCKET\_OPTION\_OOINLINE**  
*When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.*

### 6.754.1 Detailed Description

#### Since

1.0

### 6.754.2 Constructor & Destructor Documentation

6.754.2.1 virtual decaf::net::SocketOptions::~~SocketOptions ( ) [virtual]

### 6.754.3 Field Documentation

6.754.3.1 const int decaf::net::SocketOptions::SOCKET\_OPTION\_BINDADDR  
[static]

Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed).

The default local address of a socket is INADDR\_ANY, meaning any local address on a multi-homed host. A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a **ServerSocket** (p. 3292) or DatagramSocket), or to specify its return address to the peer (for a **Socket** (p. 3445) or DatagramSocket). The parameter of this option is an **InetAddress** (p. 1974).

6.754.3.2 const int decaf::net::SocketOptions::SOCKET\_OPTION\_BROADCAST  
[static]

Sets SO\_BROADCAST for a socket.

This option enables and disables the ability of the process to send broadcast messages. It is supported for only datagram sockets and only on networks that support the concept of a broadcast message (e.g. Ethernet, token ring, etc.), and it is set by default for DatagramSockets.

6.754.3.3 const int decaf::net::SocketOptions::SOCKET\_OPTION\_IP\_-  
MULTICAST\_IF [static]

Set which outgoing interface on which to send multicast packets.

Useful on hosts with multiple network interfaces, where applications want to use other than the system default. Takes/returns an **InetAddress** (p. 1974).

Valid for Multicast: DatagramSocketImpl.

6.754.3.4 const int decaf::net::SocketOptions::SOCKET\_OPTION\_IP\_-  
MULTICAST\_IF2 [static]

Same as above.

This option is introduced so that the behaviour with `IP_MULTICAST_IF` will be kept the same as before, while this new option can support setting outgoing interfaces with either IPv4 and IPv6 addresses.

**6.754.3.5** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_-  
MULTICAST_LOOP` `[static]`

This option enables or disables local loopback of multicast datagrams.

This option is enabled by default for Multicast Sockets.

**6.754.3.6** `const int decaf::net::SocketOptions::SOCKET_OPTION_IP_TOS`  
`[static]`

This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.

**6.754.3.7** `const int decaf::net::SocketOptions::SOCKET_OPTION_KEEPAIVE`  
`[static]`

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer.

This probe is a TCP segment to which the peer must respond. One of three responses is expected: 1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity. 2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed. 3. There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host crashes.

Valid only for TCP socket: **SocketImpl** (p. 3472)

**6.754.3.8** `const int decaf::net::SocketOptions::SOCKET_OPTION_LINGER`  
`[static]`

Specify a linger-on-close timeout.

This option disables/enables immediate return from a `close()` of a TCP **Socket** (p. 3445). Enabling this option with a non-zero Integer timeout means that a `close()` will block pending the transmission and acknowledgment of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully, with a TCP RST. Enabling the option with a timeout of zero does a forceful close immediately. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

Valid only for TCP: **SocketImpl** (p. 3472)

**6.754.3.9 const int decaf::net::SocketOptions::SOCKET\_OPTION\_OOBLINE**  
[static]

When the OOBLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream.

When the option is disabled (which is the default) urgent data is silently discarded.

**6.754.3.10 const int decaf::net::SocketOptions::SOCKET\_OPTION\_RCVBUF**  
[static]

Set a hint the size of the underlying buffers used by the platform for incoming network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be received over the socket. When used in get, this must return the size of the buffer actually used by the platform when receiving in data on this socket. Valid for all sockets: **SocketImpl** (p. 3472), **DatagramSocketImpl**.

**6.754.3.11 const int decaf::net::SocketOptions::SOCKET\_OPTION\_REUSEADDR**  
[static]

Sets SO\_REUSEADDR for a socket.

This is used only for MulticastSockets in decaf, and it is set by default for MulticastSockets.

**6.754.3.12 const int decaf::net::SocketOptions::SOCKET\_OPTION\_SNDBUF**  
[static]

Set a hint the size of the underlying buffers used by the platform for outgoing network I/O.

When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be sent over the socket. When used in get, this must return the size of the buffer actually used by the platform when sending out data on this socket. Valid for all sockets: **SocketImpl** (p. 3472), **DatagramSocketImpl**.

**6.754.3.13 const int decaf::net::SocketOptions::SOCKET\_OPTION\_TCP\_NODELAY**  
[static]

Disable Nagle's algorithm for this connection.

Written data to the network is not buffered pending acknowledgment of previously written data. Valid for TCP sockets.

6.754.3.14 `const int decaf::net::SocketOptions::SOCKET_OPTION_TIMEOUT`  
`[static]`

Set a timeout on blocking **Socket** (p. 3445) operations.

The option must be set prior to entering a blocking operation to take effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketOptions.h`

## 6.755 decaf::net::SocketTimeoutException Class Reference

```
#include <src/main/decaf/net/SocketTimeoutException.h>
```

Inheritance diagram for `decaf::net::SocketTimeoutException`:

### Public Member Functions

- **SocketTimeoutException** () throw ()  
*Default Constructor.*
- **SocketTimeoutException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()  
*Copy Constructor.*
- **SocketTimeoutException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **SocketTimeoutException** (const std::exception \*cause) throw ()  
*Constructor.*
- **SocketTimeoutException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **SocketTimeoutException** \* **clone** () const  
*Clones this exception.*
- virtual ~**SocketTimeoutException** () throw ()

### 6.755.1 Constructor & Destructor Documentation

6.755.1.1 `decaf::net::SocketTimeoutException::SocketTimeoutException ( ) throw ()`  
`[inline]`

Default Constructor.



6.755.1.2 `decaf::net::SocketTimeoutException::SocketTimeoutException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.755.1.3 `decaf::net::SocketTimeoutException::SocketTimeoutException ( const SocketTimeoutException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.755.1.4 `decaf::net::SocketTimeoutException::SocketTimeoutException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.755.1.5 `decaf::net::SocketTimeoutException::SocketTimeoutException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.755.1.6 `decaf::net::SocketTimeoutException::SocketTimeoutException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.755.1.7 `virtual decaf::net::SocketTimeoutException::~~SocketTimeoutException ( ) throw ()`  
`[inline, virtual]`

### 6.755.2 Member Function Documentation

6.755.2.1 `virtual SocketTimeoutException* decaf::net::SocketTimeoutException::clone ( )const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::InterruptedIOException** (p.2091).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketTimeoutException.h`

## 6.756 decaf::net::ssl::SSLContext Class Reference

Represents on implementation of the Secure **Socket** (p.3445) Layer for streaming based sockets.

```
#include <src/main/decaf/net/ssl/SSLContext.h>
```

#### Public Member Functions

- **SSLContext** (**SSLContextSpi** \*contextImpl)

- virtual `~SSLContext ()`
- **SocketFactory** \* `getSocketFactory ()`  
*Returns an **SocketFactory** (p. 3467) instance for use with this Context, the **SocketFactory** (p. 3467) is owned by the Context and should not be deleted by the caller.*
- **ServerSocketFactory** \* `getServerSocketFactory ()`  
*Returns an **ServerSocketFactory** (p. 3301) instance for use with this Context, the **ServerSocketFactory** (p. 3301) is owned by the Context and should not be deleted by the caller.*
- **SSLParameters** \* `getDefaultSSLParameters ()`
- **SSLParameters** \* `getSupportedSSLParameters ()`

### Static Public Member Functions

- static **SSLContext** \* `getDefault ()`  
*Gets the Default **SSLContext** (p. 3489).*
- static void `setDefault (SSLContext *context)`  
*Sets the default **SSLContext** (p. 3489) to be returned from future calls to `getDefault`.*

#### 6.756.1 Detailed Description

Represents on implementation of the Secure **Socket** (p.3445) Layer for streaming based sockets.

This class servers a a source of factories to be used to create new SSL **Socket** (p. 3445) instances.

#### Since

1.0

#### 6.756.2 Constructor & Destructor Documentation

6.756.2.1 `decaf::net::ssl::SSLContext::SSLContext ( SSLContextSpi * contextImpl )`

6.756.2.2 `virtual decaf::net::ssl::SSLContext::~~SSLContext ( )` [virtual]

#### 6.756.3 Member Function Documentation

6.756.3.1 `static SSLContext* decaf::net::ssl::SSLContext::getDefault ( )` [static]

Gets the Default **SSLContext** (p. 3489).

The default instance of the **SSLContext** (p. 3489) should be immediately usable without any need for the client to initialize this context.

#### Returns

a pointer to the Default **SSLContext** (p. 3489) instance.

### 6.756.3.2 **SSLParameters\*** decaf::net::ssl::SSLContext::getDefaultSSLParameters ( )

#### Returns

a new instance of an **SSLParameters** (p. 3495) object containing the default set of settings for this **SSLContext** (p. 3489).

#### Exceptions

<i>UnsupportedOperationException</i>	if the parameters cannot be retrieved.
--------------------------------------	--

### 6.756.3.3 **ServerSocketFactory\*** decaf::net::ssl::SSLContext::getServerSocketFactory ( )

Returns an **ServerSocketFactory** (p. 3301) instance for use with this Context, the **ServerSocketFactory** (p. 3301) is owned by the Context and should not be deleted by the caller.

#### Returns

a pointer to this SSLContext's **ServerSocketFactory** (p. 3301) for creating **SSLServerSocket** (p. 3498) objects.

#### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3492) requires initialization but it has not yet been initialized.
------------------------------	--

### 6.756.3.4 **SocketFactory\*** decaf::net::ssl::SSLContext::getSocketFactory ( )

Returns an **SocketFactory** (p. 3467) instance for use with this Context, the **SocketFactory** (p. 3467) is owned by the Context and should not be deleted by the caller.

#### Returns

a pointer to this SSLContext's **SocketFactory** (p. 3467) for creating **SSLSocket** (p. 3506) objects.

#### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3492) requires initialization but it has not yet been initialized.
------------------------------	--

## 6.756.3.5 SSLParameters\* decaf::net::ssl::SSLContext::getSupportedSSLParameters ( )

**Returns**

a new instance of an **SSLParameters** (p. 3495) object containing the complete set of settings for this **SSLContext** (p. 3489).

**Exceptions**

<i>UnsupportedOperationException</i>	if the parameters cannot be retrieved.
--------------------------------------	--

6.756.3.6 static void decaf::net::ssl::SSLContext::setDefault ( SSLContext \* context )  
[static]

Sets the default **SSLContext** (p. 3489) to be returned from future calls to getDefault.

The set **SSLContext** (p. 3489) must be fully initialized and usable. The caller is responsible for deleting this object before the Library shutdown methods are called.

**Exceptions**

<i>NullPointerException</i>	if the context passed is NULL.
-----------------------------	--------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLContext.h**

## 6.757 decaf::net::ssl::SSLContextSpi Class Reference

Defines the interface that should be provided by an **SSLContext** (p. 3489) provider.

```
#include <src/main/decaf/net/ssl/SSLContextSpi.h>
```

Inheritance diagram for decaf::net::ssl::SSLContextSpi:

**Public Member Functions**

- virtual ~**SSLContextSpi** ()
- virtual void **providerInit** (**security::SecureRandom** \*random)=0  
*Perform the initialization of this Context.*
- virtual **SSLParameters** \* **providerGetDefaultSSLParameters** ()  
*Creates a returns a new **SSLParameters** (p. 3495) instance that contains the default settings for this Providers **SSLContext** (p. 3489).*
- virtual **SSLParameters** \* **providerGetSupportedSSLParameters** ()

Creates and returns a new **SSLParameters** (p. 3495) instance that contains the full set of supported parameters for this **SSL Context**.

- virtual **SocketFactory** \* **providerGetSocketFactory** ()=0

Returns a **SocketFactory** (p. 3467) instance that can be used to create new **SSLSocket** (p. 3506) objects.

- virtual **ServerSocketFactory** \* **providerGetServerSocketFactory** ()=0

Returns a **ServerSocketFactory** (p. 3301) instance that can be used to create new **SSLServerSocket** (p. 3498) objects.

### 6.757.1 Detailed Description

Defines the interface that should be provided by an **SSLContext** (p. 3489) provider.

#### Since

1.0

### 6.757.2 Constructor & Destructor Documentation

6.757.2.1 virtual decaf::net::ssl::SSLContextSpi::~SSLContextSpi ( ) [virtual]

### 6.757.3 Member Function Documentation

6.757.3.1 virtual **SSLParameters**\* decaf::net::ssl::SSLContextSpi::providerGetDefaultSSLParameters ( ) [virtual]

Creates a returns a new **SSLParameters** (p. 3495) instance that contains the default settings for this Providers **SSLContext** (p. 3489).

The returned **SSLParameters** (p. 3495) instance is requires to have non-empty values in its ciphersuites and protocols.

#### Returns

new **SSLParameters** (p. 3495) instance with the **SSLContext** (p. 3489) defaults.

#### Exceptions

<i>UnsupportedOperationException</i>	if the defaults cannot be obtained.
--------------------------------------	-------------------------------------

6.757.3.2 virtual **ServerSocketFactory**\* decaf::net::ssl::SSLContextSpi::providerGetServerSocketFactory ( ) [pure virtual]

Returns a **ServerSocketFactory** (p. 3301) instance that can be used to create new **SSLServerSocket** (p. 3498) objects.

The **ServerSocketFactory** (p. 3301) is owned by the Service Provider and should not be destroyed by the caller.

### Returns

**SocketFactory** (p. 3467) instance that can be used to create new SSLServerSockets.

### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3492) object requires initialization but has not been initialized yet.
------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2794).

**6.757.3.3** virtual **SocketFactory\*** decaf::net::ssl::SSLContextSpi::providerGetSocketFactory ( ) [pure virtual]

Returns a **SocketFactory** (p. 3467) instance that can be used to create new **SSLSocket** (p. 3506) objects.

The **SocketFactory** (p. 3467) is owned by the Service Provider and should not be destroyed by the caller.

### Returns

**SocketFactory** (p. 3467) instance that can be used to create new SSLSockets.

### Exceptions

<i>IllegalStateException</i>	if the <b>SSLContextSpi</b> (p. 3492) object requires initialization but has not been initialized yet.
------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLContextSpi** (p. 2794).

**6.757.3.4** virtual **SSLParameters\*** decaf::net::ssl::SSLContextSpi::providerGetSupportedSSLParameters ( ) [virtual]

Creates and returns a new **SSLParameters** (p. 3495) instance that contains the full set of supported parameters for this SSL Context.

The returned **SSLParameters** (p. 3495) instance is requires to have non-empty values in its ciphersuites and protocols.

### Returns

a new **SSLParameters** (p. 3495) instance with the full set of settings that are supported.

### Exceptions

<i>UnsupportedOperationException</i>	if the supported parameters cannot be obtained.
--------------------------------------	---

6.757.3.5 `virtual void decaf::net::ssl::SSLContextSpi::providerInit ( security::SecureRandom * random ) [pure virtual]`

Perform the initialization of this Context.

#### Parameters

<i>random</i>	Pointer to an instance of a secure random number generator.
---------------	---

#### Exceptions

<i>NullPointerException</i>	if the SecureRandom instance is NULL.
<i>KeyManagementException</i>	if an error occurs while initializing the context.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLContextSpi` (p. 2794).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLContextSpi.h`

## 6.758 decaf::net::ssl::SSLParameters Class Reference

```
#include <src/main/decaf/net/ssl/SSLParameters.h>
```

### Public Member Functions

- **SSLParameters ()**  
*Creates a new **SSLParameters** (p. 3495) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.*
- **SSLParameters** (const std::vector< std::string > &cipherSuites)  
*Creates a new **SSLParameters** (p. 3495) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.*
- **SSLParameters** (const std::vector< std::string > &cipherSuites, const std::vector< std::string > &protocols)  
*Creates a new **SSLParameters** (p. 3495) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.*
- virtual **~SSLParameters ()**
- std::vector< std::string > **getCipherSuites ()** const
- void **setCipherSuites** (const std::vector< std::string > &cipherSuites)  
*Sets the vector of ciphersuites.*



- `std::vector< std::string > getProtocols () const`
- `void setProtocols (const std::vector< std::string > &protocols)`  
*Sets the vector of protocols.*
- `bool getWantClientAuth () const`
- `void setWantClientAuth (bool wantClientAuth)`  
*Sets whether client authentication should be requested.*
- `bool getNeedClientAuth () const`
- `void setNeedClientAuth (bool needClientAuth)`  
*Sets whether client authentication should be required.*

### 6.758.1 Constructor & Destructor Documentation

#### 6.758.1.1 `decaf::net::ssl::SSLParameters::SSLParameters ( )`

Creates a new **SSLParameters** (p. 3495) instance with empty vectors for the protocols and the cipherSuites, the wantClientAuth and needClientAuth flags are set to false.

#### 6.758.1.2 `decaf::net::ssl::SSLParameters::SSLParameters ( const std::vector< std::string > & cipherSuites )`

Creates a new **SSLParameters** (p. 3495) instance with the given cipherSuites value, the protocols vector is empty and the wantClientAuth and needClientAuth flags are set to false.

##### Parameters

<i>cipherSuites</i>	The vector of cipherSuites for this <b>SSLParameters</b> (p. 3495) instance (can be empty).
---------------------	---

#### 6.758.1.3 `decaf::net::ssl::SSLParameters::SSLParameters ( const std::vector< std::string > & cipherSuites, const std::vector< std::string > & protocols )`

Creates a new **SSLParameters** (p. 3495) instance with the given cipherSuites value and protocols value, the wantClientAuth and needClientAuth flags are set to false.

##### Parameters

<i>cipherSuites</i>	The vector of cipherSuites for this <b>SSLParameters</b> (p. 3495) instance (can be empty).
<i>protocols</i>	The vector of protocols for this <b>SSLParameters</b> (p. 3495) instance (can be empty).

#### 6.758.1.4 `virtual decaf::net::ssl::SSLParameters::~~SSLParameters ( ) [virtual]`

## 6.758.2 Member Function Documentation

6.758.2.1 `std::vector<std::string> decaf::net::ssl::SSLParameters::getCipherSuites ( ) const`  
[inline]

### Returns

a copy of the vector of ciphersuites or an empty vector if none have been set.

6.758.2.2 `bool decaf::net::ssl::SSLParameters::getNeedClientAuth ( ) const` [inline]

### Returns

whether client authentication should be required.

6.758.2.3 `std::vector<std::string> decaf::net::ssl::SSLParameters::getProtocols ( ) const`  
[inline]

### Returns

a copy of the vector of protocols or an empty vector if none have been set.

6.758.2.4 `bool decaf::net::ssl::SSLParameters::getWantClientAuth ( ) const` [inline]

### Returns

whether client authentication should be requested.

6.758.2.5 `void decaf::net::ssl::SSLParameters::setCipherSuites ( const std::vector< std::string  
> & cipherSuites )` [inline]

Sets the vector of ciphersuites.

### Parameters

<i>cipherSuites</i>	The vector of cipherSuites (can be an empty vector).
---------------------	--

6.758.2.6 `void decaf::net::ssl::SSLParameters::setNeedClientAuth ( bool needClientAuth )`  
[inline]

Sets whether client authentication should be required.

Calling this method clears the wantClientAuth flag.

**Parameters**

<i>needClientAuth</i>	whether client authentication should be required.
-----------------------	---

6.758.2.7 void decaf::net::ssl::SSLParameters::setProtocols ( const std::vector< std::string > & protocols ) [inline]

Sets the vector of protocols.

**Parameters**

<i>protocols</i>	the vector of protocols (or an empty vector)
------------------	--

6.758.2.8 void decaf::net::ssl::SSLParameters::setWantClientAuth ( bool wantClientAuth ) [inline]

Sets whether client authentication should be requested.

Calling this method clears the needClientAuth flag.

**Parameters**

<i>whether</i>	client authentication should be requested.
----------------	--

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/SSLParameters.h

**6.759 decaf::net::ssl::SSLServerSocket Class Reference**

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

```
#include <src/main/decaf/net/ssl/SSLServerSocket.h>
```

Inheritance diagram for decaf::net::ssl::SSLServerSocket:

**Public Member Functions**

- virtual ~SSLServerSocket ()
- virtual std::vector< std::string > getSupportedCipherSuites () const =0  
*Gets a vector containing the names of all the cipher suites that are supported by this SSLServerSocket (p. 3498).*

- virtual std::vector< std::string > **getSupportedProtocols** () const =0  
*Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3498) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0  
*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3498).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0  
*Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3498) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0  
*Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3498).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0  
*Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3498) connection.*
- virtual bool **getWantClientAuth** () const =0
- virtual void **setWantClientAuth** (bool value)=0  
*Sets whether or not this **Socket** (p. 3445) will request Client Authentication.*
- virtual bool **getNeedClientAuth** () const =0
- virtual void **setNeedClientAuth** (bool value)=0  
*Sets whether or not this **Socket** (p. 3445) will require Client Authentication.*

## Protected Member Functions

- **SSLServerSocket** ()  
*Creates a non-bound server socket.*
- **SSLServerSocket** (int port)  
*Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **SSLServerSocket** (int port, int backlog)  
*Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.*
- **SSLServerSocket** (int port, int backlog, const **decaf::net::InetAddress** \*address)  
*Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.*

### 6.759.1 Detailed Description

Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.

The main function of this class is to create **SSLSocket** (p. 3506) objects by accepting connections from client sockets over SSL.

**Since**

1.0

**6.759.2 Constructor & Destructor Documentation****6.759.2.1 decaf::net::ssl::SSLServerSocket::SSLServerSocket ( )** [protected]

Creates a non-bound server socket.

**6.759.2.2 decaf::net::ssl::SSLServerSocket::SSLServerSocket ( int *port* )** [protected]

Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at 50, connections that arrive after the backlog has been reached are refused.

If a **SocketImplFactory** (p. 3481) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3472) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
-------------	--

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentException</i>	if the port value is negative or greater than 65535.

**6.759.2.3 decaf::net::ssl::SSLServerSocket::SSLServerSocket ( int *port*, int *backlog* )** [protected]

Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.

When this constructor is called the size of the backlog queue is set at backlog, connections that arrive after the backlog has been reached are refused. If backlog is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3481) is registered then the createSocketImpl method on the factory will be called otherwise a default **SocketImpl** (p. 3472) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentEx- ception</i>	if the port value is negative or greater than 65535.

**6.759.2.4** `decaf::net::ssl::SSLServerSocket::SSLServerSocket ( int port, int backlog, const decaf::net::InetAddress * address )` `[protected]`

Creates a new **ServerSocket** (p. 3292) bound to the specified port, if the value of port is 0, then any free port is chosen.

If the value of the *ifAddress* is empty or NULL then the ANY address is used.

When this constructor is called the size of the backlog queue is set at *backlog*, connections that arrive after the backlog has been reached are refused. If *backlog* is zero or negative then the default backlog value of 50 is used.

If a **SocketImplFactory** (p. 3481) is registered then the `createSocketImpl` method on the factory will be called otherwise a default **SocketImpl** (p. 3472) is created.

**Parameters**

<i>port</i>	The port to bind the <b>ServerSocket</b> (p. 3292) to.
<i>backlog</i>	The the number of incoming connection attempts to queue before connections are refused.
<i>ifAddress</i>	The IP Address to bind to on the local machine.

**Exceptions**

<i>IOException</i>	if there is an I/O error while performing this operation.
<i>IllegalArgumentEx- ception</i>	if the port value is negative or greater than 65535.

**6.759.2.5** `virtual decaf::net::ssl::SSLServerSocket::~SSLServerSocket ( )` `[virtual]`

**6.759.3 Member Function Documentation**

**6.759.3.1** `virtual std::vector<std::string> decaf::net::ssl::SSLServerSocket::getEnabledCipherSuites ( )`  
`const` `[pure virtual]`

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSLServerSocket** (p. 3498).

**Returns**

vector of the names of all enabled Cipher Suites.

Implemented in `decaf::internal::net::ssl::openssl::OpenSSLServerSocket` (p. 2800).

```
6.759.3.2  virtual std::vector<std::string> de-
           caf::net::ssl::SSLServerSocket::getEnabledProtocols ( ) const
           [pure virtual]
```

Returns a vector containing the names of all the currently enabled Protocols for this **SSLServerSocket** (p. 3498).

#### Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2800).

```
6.759.3.3  virtual bool decaf::net::ssl::SSLServerSocket::getNeedClientAuth ( ) const  [pure
           virtual]
```

#### Returns

true if the **Socket** (p. 3445) requires client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2800).

```
6.759.3.4  virtual std::vector<std::string> de-
           caf::net::ssl::SSLServerSocket::getSupportedCipherSuites ( )
           const  [pure virtual]
```

Gets a vector containing the names of all the cipher suites that are supported by this **SSLServerSocket** (p. 3498).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3445).

#### Returns

a vector containing the names of all the supported cipher suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2801).

```
6.759.3.5  virtual std::vector<std::string> de-
           caf::net::ssl::SSLServerSocket::getSupportedProtocols ( )
           const  [pure virtual]
```

Gets a vector containing the names of all the protocols that could be enabled for this **SSLServerSocket** (p. 3498) instance.

#### Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2801).

6.759.3.6 `virtual bool decaf::net::ssl::SSLServerSocket::getWantClientAuth ( ) const` [pure virtual]

#### Returns

true if the **Socket** (p. 3445) request client Authentication.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2801).

6.759.3.7 `virtual void decaf::net::ssl::SSLServerSocket::setEnabledCipherSuites ( const std::vector< std::string > & suites )` [pure virtual]

Sets the Cipher Suites that are to be enabled on the **SSLServerSocket** (p. 3498) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

#### Parameters

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if the vector is empty or one of the names is invalid.
---------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2801).

6.759.3.8 `virtual void decaf::net::ssl::SSLServerSocket::setEnabledProtocols ( const std::vector< std::string > & protocols )` [pure virtual]

Sets the Protocols that are to be enabled on the **SSLServerSocket** (p. 3498) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

#### Parameters

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if the vector is empty or one of the names is invalid.
---------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2802).



6.759.3.9 virtual void decaf::net::ssl::SSLServerSocket::setNeedClientAuth ( bool *value* )  
[pure virtual]

Sets whether or not this **Socket** (p. 3445) will require Client Authentication.

If set to true the **Socket** (p. 3445) (when used in server mode) will require that the client authenticate itself, if the client doesn't send authentication the socket will not allow negotiation to continue.

#### Parameters

<i>value</i>	Whether the server socket should require client authentication.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2802).

6.759.3.10 virtual void decaf::net::ssl::SSLServerSocket::setWantClientAuth ( bool *value* )  
[pure virtual]

Sets whether or not this **Socket** (p. 3445) will request Client Authentication.

If set to true the **Socket** (p. 3445) (when used in server mode) will request that the client authenticate itself, if the client doesn't send authentication the socket will still allow negotiation to continue.

#### Parameters

<i>value</i>	Whether the server socket should request client authentication.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLServerSocket** (p. 2802).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLServerSocket.h**

## 6.760 decaf::net::ssl::SSLServerSocketFactory Class Reference

Factory class interface that provides methods to create SSL Server Sockets.

```
#include <src/main/decaf/net/ssl/SSLServerSocketFactory.h>
```

Inheritance diagram for decaf::net::ssl::SSLServerSocketFactory:

#### Public Member Functions

- virtual ~**SSLServerSocketFactory** ()
- virtual std::vector< std::string > **getDefaultCipherSuites** ()=0  
*Returns the list of cipher suites which are enabled by default.*

- virtual std::vector< std::string > **getSupportedCipherSuites** ()=0

*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*

### Static Public Member Functions

- static **ServerSocketFactory** \* **getDefault** ()

*Returns the current default SSL **ServerSocketFactory** (p. 3301), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

### Protected Member Functions

- **SSLServerSocketFactory** ()

#### 6.760.1 Detailed Description

Factory class interface that provides methods to create SSL Server Sockets.

#### Since

1.0

#### 6.760.2 Constructor & Destructor Documentation

6.760.2.1 **decaf::net::ssl::SSLServerSocketFactory::SSLServerSocketFactory** ( )  
[protected]

6.760.2.2 **virtual decaf::net::ssl::SSLServerSocketFactory::~~SSLServerSocketFactory** ( )  
[virtual]

#### 6.760.3 Member Function Documentation

6.760.3.1 **static ServerSocketFactory\* decaf::net::ssl::SSLServerSocketFactory::getDefault** ( ) [static]

Returns the current default SSL **ServerSocketFactory** (p. 3301), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 3490)->**getServerSocketFactory()**. If that call fails, a non-functional factory is returned.

#### Returns

the default SSL **ServerSocketFactory** (p. 3301) pointer.

**See also**

**decaf::net::ssl::SSLContext::getDefault()** (p. 3490)

Reimplemented from **decaf::net::ServerSocketFactory** (p. 3304).

```
6.760.3.2  virtual std::vector<std::string> de-
            caf::net::ssl::SSLServerSocketFactory::getDefaultCipherSuites (
            ) [pure virtual]
```

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

**getSupportedCipherSuites()** (p. 3506)

Implemented in **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1662), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2807).

```
6.760.3.3  virtual std::vector<std::string> de-
            caf::net::ssl::SSLServerSocketFactory::getSupportedCipherSuites
            ( ) [pure virtual]
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

**getDefaultCipherSuites()** (p. 3505)

Implemented in **decaf::internal::net::ssl::DefaultSSLServerSocketFactory** (p. 1662), and **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory** (p. 2807).

The documentation for this class was generated from the following file:

- src/main/decaf/net/ssl/**SSLServerSocketFactory.h**

## 6.761 decaf::net::ssl::SSLSocket Class Reference

```
#include <src/main/decaf/net/ssl/SSLSocket.h>
```

Inheritance diagram for decaf::net::ssl::SSLSocket:

### Public Member Functions

- **SSLSocket** ()
- **SSLSocket** (const **InetAddress** \*address, int port)
 

*Creates a new **SSLSocket** (p. 3506) instance and connects it to the given address and port.*
- **SSLSocket** (const **InetAddress** \*address, int port, const **InetAddress** \*localAddress, int localPort)
 

*Creates a new **SSLSocket** (p. 3506) instance and connects it to the given address and port.*
- **SSLSocket** (const std::string &host, int port)
 

*Creates a new **SSLSocket** (p. 3506) instance and connects it to the given host and port.*
- **SSLSocket** (const std::string &host, int port, const **InetAddress** \*localAddress, int localPort)
 

*Creates a new **SSLSocket** (p. 3506) instance and connects it to the given host and port.*
- virtual ~**SSLSocket** ()
- virtual std::vector< std::string > **getSupportedCipherSuites** () const =0
 

*Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3506).*
- virtual std::vector< std::string > **getSupportedProtocols** () const =0
 

*Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3506) instance.*
- virtual std::vector< std::string > **getEnabledCipherSuites** () const =0
 

*Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3445).*
- virtual void **setEnabledCipherSuites** (const std::vector< std::string > &suites)=0
 

*Sets the Cipher Suites that are to be enabled on the **SSL Socket** (p. 3445) connection.*
- virtual std::vector< std::string > **getEnabledProtocols** () const =0
 

*Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3445).*
- virtual void **setEnabledProtocols** (const std::vector< std::string > &protocols)=0
 

*Sets the Protocols that are to be enabled on the **SSL Socket** (p. 3445) connection.*
- virtual **SSLParameters** **getSSLParameters** () const
 

*Returns an **SSLParameters** (p. 3495) object for this **SSLSocket** (p. 3506) instance.*

- virtual void **setSSLParameters** (const **SSLParameters** &value)  
*Sets the **SSLParameters** (p. 3495) for this **SSLSocket** (p. 3506) using the supplied **SSLParameters** (p. 3495) instance.*
- virtual void **startHandshake** ()=0  
*Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.*
- virtual void **setUseClientMode** (bool value)=0  
*Determines the mode that the socket uses when a handshake is initiated, client or server.*
- virtual bool **getUseClientMode** () const =0  
*Gets whether this **Socket** (p. 3445) is in Client or Server mode, true indicates that the mode is set to Client.*
- virtual void **setNeedClientAuth** (bool value)=0  
*Sets the **Socket** (p. 3445) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getNeedClientAuth** () const =0  
*Returns if this socket is configured to require client authentication, true means that is has and that clients that failed to authenticate will be rejected.*
- virtual void **setWantClientAuth** (bool value)=0  
*Sets the **Socket** (p. 3445) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.*
- virtual bool **getWantClientAuth** () const =0  
*Returns if this socket is configured to request client authentication, true means that is has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.*

### 6.761.1 Detailed Description

Since

1.0

### 6.761.2 Constructor & Destructor Documentation

6.761.2.1 decaf::net::ssl::SSLSocket::SSLSocket ( )

6.761.2.2 decaf::net::ssl::SSLSocket::SSLSocket ( const InetAddress \* address, int port )

Creates a new **SSLSocket** (p. 3506) instance and connects it to the given address and port.

If the host parameter is empty then the loop back address is used.

#### Parameters

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]

**Exceptions**

<b><i>UnknownHostException</i></b> (p. 3841)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).
<i>NullPointerException</i>	if the <b>InetAddress</b> (p. 1974) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

6.761.2.3 `decaf::net::ssl::SSLSocket::SSLSocket ( const InetAddress * address, int port, const InetAddress * localAddress, int localPort )`

Creates a new **SSLSocket** (p. 3506) instance and connects it to the given address and port.

The **Socket** (p. 3445) will also bind to the local address and port specified.

**Parameters**

<i>address</i>	The address to connect to.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

**Exceptions**

<b><i>UnknownHostException</i></b> (p. 3841)	if the host cannot be resolved.
<i>IOException</i>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).
<i>NullPointerException</i>	if the <b>InetAddress</b> (p. 1974) instance is NULL.
<i>IllegalArgumentException</i>	if the port is not in range [0...65535]

6.761.2.4 `decaf::net::ssl::SSLSocket::SSLSocket ( const std::string & host, int port )`

Creates a new **SSLSocket** (p. 3506) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

**Parameters**

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]

**Exceptions**

<b>UnknownHostException</b> (p. 3841)	if the host cannot be resolved.
<b>IOException</b>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).
<b>IllegalArgumentException</b>	if the port if not in range [0...65535]

6.761.2.5 decaf::net::ssl::SSLSocket::SSLSocket ( const std::string & *host*, int *port*, const InetAddress \* *localAddress*, int *localPort* )

Creates a new **SSLSocket** (p. 3506) instance and connects it to the given host and port.

If the host parameter is empty then the loop back address is used.

#### Parameters

<i>host</i>	The host name or IP address to connect to, empty string means loopback.
<i>port</i>	The port number to connect to [0...65535]
<i>localAddress</i>	The IP address on the local machine to bind to.
<i>localPort</i>	The port on the local machine to bind to.

#### Exceptions

<b>UnknownHostException</b> (p. 3841)	if the host cannot be resolved.
<b>IOException</b>	if an I/O error occurs while connecting the <b>Socket</b> (p. 3445).
<b>IllegalArgumentException</b>	if the port if not in range [0...65535]

6.761.2.6 virtual decaf::net::ssl::SSLSocket::~~SSLSocket ( ) [virtual]

### 6.761.3 Member Function Documentation

6.761.3.1 virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledCipherSuites ( ) const [pure virtual]

Returns a vector containing the names of all the currently enabled Cipher Suites for this **SSL Socket** (p. 3445).

#### Returns

vector of the names of all enabled Cipher Suites.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSSLSocket** (p. 2814).

```
6.761.3.2  virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getEnabledProtocols ( )  
          const [pure virtual]
```

Returns a vector containing the names of all the currently enabled Protocols for this **SSL Socket** (p. 3445).

#### Returns

vector of the names of all enabled Protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2814).

```
6.761.3.3  virtual bool decaf::net::ssl::SSLSocket::getNeedClientAuth ( ) const [pure  
          virtual]
```

Returns if this socket is configured to require client authentication, true means that it has and that clients that failed to authenticate will be rejected.

This option is only useful when the socket is operating in server mode.

#### Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2815).

```
6.761.3.4  virtual SSLParameters decaf::net::ssl::SSLSocket::getSSLParameters ( ) const  
          [virtual]
```

Returns an **SSLParameters** (p. 3495) object for this **SSLSocket** (p. 3506) instance.

The cipherSuites and protocols vectors in the returned **SSLParameters** (p. 3495) reference will never be empty.

#### Returns

an **SSLParameters** (p. 3495) object with the settings in use for the **SSLSocket** (p. 3506).

```
6.761.3.5  virtual std::vector<std::string> de-  
          caf::net::ssl::SSLSocket::getSupportedCipherSuites ( ) const  
          [pure virtual]
```

Gets a vector containing the names of all the cipher suites that are supported by this **SSLSocket** (p. 3506).

Normally not all of these cipher suites will be enabled on the **Socket** (p. 3445).

#### Returns

a vector containing the names of all the supported cipher suites.



Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2816).

6.761.3.6 `virtual std::vector<std::string> decaf::net::ssl::SSLSocket::getSupportedProtocols ( ) const [pure virtual]`

Gets a vector containing the names of all the protocols that could be enabled for this **SSLSocket** (p. 3506) instance.

#### Returns

a vector containing the names of all the supported protocols.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2816).

6.761.3.7 `virtual bool decaf::net::ssl::SSLSocket::getUseClientMode ( ) const [pure virtual]`

Gets whether this **Socket** (p. 3445) is in Client or Server mode, true indicates that the mode is set to Client.

#### Returns

true if the **Socket** (p. 3445) is in Client mode, false otherwise.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2816).

6.761.3.8 `virtual bool decaf::net::ssl::SSLSocket::getWantClientAuth ( ) const [pure virtual]`

Returns if this socket is configured to request client authentication, true means that it has and that clients that failed to authenticate will be rejected but that clients that do not send a certificate are not considered to have failed authentication.

This option is only useful when the socket is operating in server mode.

#### Returns

true if client authentication is required.

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2817).

6.761.3.9 `virtual void decaf::net::ssl::SSLSocket::setEnabledCipherSuites ( const std::vector<std::string> & suites ) [pure virtual]`

Sets the Cipher Suites that are to be enabled on the SSL **Socket** (p. 3445) connection.

Each of the named Cipher Suites must appear in the list of supported cipher suites for this connection or an exception will be thrown.

**Parameters**

<i>suites</i>	An Vector of names for all the Cipher Suites that are to be enabled.
---------------	--

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2818).

6.761.3.10 `virtual void decaf::net::ssl::SSLSocket::setEnabledProtocols ( const std::vector< std::string > & protocols ) [pure virtual]`

Sets the Protocols that are to be enabled on the SSL **Socket** (p. 3445) connection.

Each of the named Protocols must appear in the list of supported protocols suites for this connection or an exception will be thrown.

**Parameters**

<i>protocols</i>	An Vector of names for all the Protocols that are to be enabled.
------------------	--

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the vector is empty or one of the names is invalid.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2818).

6.761.3.11 `virtual void decaf::net::ssl::SSLSocket::setNeedClientAuth ( bool value ) [pure virtual]`

Sets the **Socket** (p. 3445) to require that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled an the client does not provide a certificate then the handshake is considered failed and the connection is refused. Calling this method resets any previous value for this option as well as clears any value set in the setWantClientAuth method.

**Parameters**

<i>value</i>	The value indicating if a client is required to authenticate itself or not.
--------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2818).

**6.761.3.12** virtual void decaf::net::ssl::SSLSocket::setSSLParameters ( const SSLParameters & value ) [virtual]

Sets the **SSLParameters** (p. 3495) for this **SSLSocket** (p. 3506) using the supplied **SSLParameters** (p. 3495) instance.

If the cipherSuites vector in the **SSLParameters** (p. 3495) instance is not empty then the setEnabledCipherSuites method is called with that vector, if the protocols vector in the **SSLParameters** (p. 3495) instance is not empty then the setEnabledProtocols method is called with that vector. If the needClientAuth value or the wantClientAuth value is true then the setNeedClientAuth and setWantClientAuth methods are called respectively with a value of true, otherwise the setWantClientAuth method is called with a value of false.

#### Parameters

value	The <b>SSLParameters</b> (p. 3495) instance that is used to update this SSLSocket's settings.
-------	---

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if an error occurs while calling setEnabledCipherSuites or setEnabledProtocols.
---------------------------------------	---

**6.761.3.13** virtual void decaf::net::ssl::SSLSocket::setUseClientMode ( bool value ) [pure virtual]

Determines the mode that the socket uses when a handshake is initiated, client or server.

This method must be called prior to any handshake attempts on this **Socket** (p. 3445), once a handshake has been initiated this socket remains in the set mode; client or server, for the life of this object.

#### Parameters

value	The mode setting, true for client or false for server.
-------	--

#### Exceptions

<i>IllegalArgumentEx- ception</i>	if the handshake process has begun and mode is locked.
---------------------------------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2819).

**6.761.3.14** virtual void decaf::net::ssl::SSLSocket::setWantClientAuth ( bool value ) [pure virtual]

Sets the **Socket** (p. 3445) to request that a client authenticate itself by sending a valid Certificate that is trusted by this Server mode socket.

This option only applies to sockets in the Server mode.

If the option is enabled and the client does not provide a certificate then the handshake is considered to have succeeded, if it does send a certificate and that certificate is invalid the the handshake will fail. Calling this method resets any previous value for this option as well as clears any value set in the `setNeedClientAuth` method.

#### Parameters

<i>value</i>	The value indicating if a client is requested to authenticate itself or not.
--------------	--

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2820).

**6.761.3.15** `virtual void decaf::net::ssl::SSLSocket::startHandshake ( ) [pure virtual]`

Initiates a handshake for this SSL Connection, this can be necessary for several reasons such as using new encryption keys, or starting a new session.

When called for the first time after the socket connects this method blocks until the handshake is completed. The provider is not required to support multiple handshakes and can throw an `IOException` to indicate an error.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing the Handshake
--------------------	---

Implemented in **decaf::internal::net::ssl::openssl::OpenSSLSocket** (p. 2820).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocket.h`

## 6.762 decaf::net::ssl::SSLSocketFactory Class Reference

Factory class interface for a **SocketFactory** (p. 3467) that can create **SSLSocket** (p. 3506) objects.

```
#include <src/main/decaf/net/ssl/SSLSocketFactory.h>
```

Inheritance diagram for `decaf::net::ssl::SSLSocketFactory`:

#### Public Member Functions

- `virtual ~SSLSocketFactory ()`
- `virtual std::vector< std::string > getDefaultCipherSuites ()=0`  
*Returns the list of cipher suites which are enabled by default.*

- virtual `std::vector< std::string > getSupportedCipherSuites ()=0`  
*Returns the names of the cipher suites which could be enabled for use on an SSL connection.*
- virtual `Socket * createSocket (Socket *socket, std::string host, int port, bool autoClose)=0`  
*Returns a socket layered over an existing socket connected to the named host, at the given port.*

### Static Public Member Functions

- static `SocketFactory * getDefault ()`  
*Returns the current default SSL **SocketFactory** (p. 3467), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.*

### Protected Member Functions

- `SSLSocketFactory ()`

#### 6.762.1 Detailed Description

Factory class interface for a **SocketFactory** (p. 3467) that can create **SSLSocket** (p. 3506) objects.

#### Since

1.0

#### 6.762.2 Constructor & Destructor Documentation

6.762.2.1 `decaf::net::ssl::SSLSocketFactory::SSLSocketFactory ( )` [protected]

6.762.2.2 `virtual decaf::net::ssl::SSLSocketFactory::~~SSLSocketFactory ( )` [virtual]

#### 6.762.3 Member Function Documentation

6.762.3.1 `virtual Socket* decaf::net::ssl::SSLSocketFactory::createSocket ( Socket * socket, std::string host, int port, bool autoClose )` [pure virtual]

Returns a socket layered over an existing socket connected to the named host, at the given port.

This constructor can be used when tunneling SSL through a proxy or when negotiating the use of SSL over an existing socket. The host and port refer to the logical peer destination. This socket is configured using the socket options established for this factory.

#### Parameters

<i>socket</i>	The existing socket to layer over.
<i>host</i>	The server host the original <b>Socket</b> (p. 3445) is connected to.
<i>port</i>	The server port the original <b>Socket</b> (p. 3445) is connected to.
<i>autoClose</i>	Should the layered over <b>Socket</b> (p. 3445) be closed when the topmost socket is closed.

**Returns**

a new **Socket** (p. 3445) instance that wraps the given **Socket** (p. 3445).

**Exceptions**

<i>IOException</i>	if an I/O exception occurs while performing this operation.
<b>UnknownHostException</b> (p. 3841)	if the host is unknown.

Implemented in **decaf::internal::net::ssl::DefaultSSLSocketFactory** (p. 1666), and **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory** (p. 2828).

6.762.3.2 **static SocketFactory\* decaf::net::ssl::SSLSocketFactory::getDefault ( )**  
[static]

Returns the current default SSL **SocketFactory** (p. 3467), the factory is returned as a pointer however the caller does not own this pointer and should not delete it.

This method returns **SSLContext::getDefault()** (p. 3490)->getSocketFactory(). If that call fails, a non-functional factory is returned.

**Returns**

the default SSL **SocketFactory** (p. 3467) pointer.

**See also**

**decaf::net::ssl::SSLContext::getDefault()** (p. 3490)

Reimplemented from **decaf::net::SocketFactory** (p. 3471).

6.762.3.3 **virtual std::vector<std::string> decaf::net::ssl::SSLSocketFactory::getDefaultCipherSuites ( )**  
[pure virtual]

Returns the list of cipher suites which are enabled by default.

Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication (that is, no anonymous cipher suites).

**Returns**

an STL vector containing the list of cipher suites enabled by default.

**See also**

`getSupportedCipherSuites()` (p. 3517)

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p. 1669), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p. 2831).

```
6.762.3.4 virtual std::vector<std::string> de-
caf::net::ssl::SSLSocketFactory::getSupportedCipherSuites ( )
[pure virtual]
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection.

Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

**Returns**

an STL vector containing the list of supported cipher suites.

**See also**

`getDefaultCipherSuites()` (p. 3517)

Implemented in `decaf::internal::net::ssl::DefaultSSLSocketFactory` (p. 1670), and `decaf::internal::net::ssl::openssl::OpenSSLSocketFactory` (p. 2832).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ssl/SSLSocketFactory.h`

**6.763 `activemq::transport::tcp::SslTransport` Class Reference**

**Transport** (p. 3819) for connecting to a Broker using an SSL Socket.

```
#include <src/main/activemq/transport/tcp/SslTransport.h>
```

Inheritance diagram for `activemq::transport::tcp::SslTransport`:

**Public Member Functions**

- **SslTransport** (const **Pointer**< **Transport** > &next)

Creates a new instance of the **SslTransport** (p. 3518), the transport will not attempt to connect to a remote host until the connect method is called.

- virtual **~SslTransport** ()

## Protected Member Functions

- virtual **decaf::net::Socket \* createSocket** ()

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

### Returns

a newly created unconnected Socket instance.

### Exceptions

IOException	if there is an error while creating the unconnected Socket.
-------------	---

- virtual void **configureSocket** (decaf::net::Socket \*socket, decaf::util::Properties &properties)

## 6.763.1 Detailed Description

**Transport** (p. 3819) for connecting to a Broker using an SSL Socket.

This transport simply wraps the **TcpTransport** (p. 3696) and provides the **TcpTransport** (p. 3696) an SSL based Socket pointer allowing the core **TcpTransport** (p. 3696) logic to be reused.

### Since

3.2.0

## 6.763.2 Constructor & Destructor Documentation

- 6.763.2.1 **activemq::transport::tcp::SslTransport::SslTransport** ( const Pointer< Transport > & next )

Creates a new instance of the **SslTransport** (p. 3518), the transport will not attempt to connect to a remote host until the connect method is called.

### Parameters

next	the next transport in the chain
------	---------------------------------

- 6.763.2.2 **virtual activemq::transport::tcp::SslTransport::~~SslTransport** ( ) [virtual]

## 6.763.3 Member Function Documentation



6.763.3.1 virtual void activemq::transport::tcp::SslTransport::configureSocket (   
**decaf::net::Socket** \* *socket*, **decaf::util::Properties** & *properties* )   
 [protected, virtual]

6.763.3.2 virtual **decaf::net::Socket**\* activemq::transport::tcp::SslTransport::createSocket (   
 ) [protected, virtual]

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

### Returns

a newly created unconnected Socket instance.

### Exceptions

<i>IOException</i>	if there is an error while creating the unconnected Socket.
--------------------	---

Reimplemented from **activemq::transport::tcp::TcpTransport** (p. 3698).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**SslTransport.h**

## 6.764 activemq::transport::tcp::SslTransportFactory Class Reference

```
#include <src/main/activemq/transport/tcp/SslTransportFactory.h>
```

Inheritance diagram for activemq::transport::tcp::SslTransportFactory:

### Public Member Functions

- virtual ~**SslTransportFactory** ()

### Protected Member Functions

- virtual **Pointer**< **Transport** > **doCreateComposite** (const **decaf::net::URI** &location, const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **decaf::util::Properties** &properties) throw ( exceptions::ActiveMQException )

*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

### 6.764.1 Constructor & Destructor Documentation

6.764.1.1 virtual `activemq::transport::tcp::SslTransportFactory::~SslTransportFactory ( )`  
`[virtual]`

## 6.764.2 Member Function Documentation

6.764.2.1 virtual `Pointer<Transport> activemq::transport::tcp::SslTransportFactory::doCreateComposite ( const decaf::net::URI & location, const Pointer<wireformat::WireFormat> & wireFormat, const decaf::util::Properties & properties ) throw ( exceptions::ActiveMQException )` `[protected, virtual]`

Creates a slimed down **Transport** (p.3819) instance which can be used in composite transport instances.

### Parameters

<i>location</i>	- URI location to connect to.
<i>wireFormat</i>	- the assigned WireFormat for the new <b>Transport</b> (p.3819).
<i>properties</i>	- Properties to apply to the transport.

### Returns

new Pointer to a **SslTransport** (p.3518).

### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Reimplemented from `activemq::transport::tcp::TcpTransportFactory` (p.3701).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/SslTransportFactory.h`

## 6.765 `activemq::commands::BrokerError::StackTraceElement` Struct Reference

```
#include <src/main/activemq/commands/BrokerError.h>
```

### Data Fields

- `std::string` **ClassName**
- `std::string` **FileName**
- `std::string` **MethodName**
- `int` **LineNumber**

### 6.765.1 Field Documentation

6.765.1.1 `std::string activemq::commands::BrokerError::StackTraceElement::ClassName`

6.765.1.2 `std::string activemq::commands::BrokerError::StackTraceElement::FileName`

6.765.1.3 `int activemq::commands::BrokerError::StackTraceElement::LineNumber`

6.765.1.4 `std::string activemq::commands::BrokerError::StackTraceElement::MethodName`

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

## 6.766 decaf::internal::io::StandardErrorOutputStream Class Reference

Wrapper Around the Standard error Output facility on the current platform.

```
#include <src/main/decaf/internal/io/StandardErrorOutputStream.h>
```

Inheritance diagram for `decaf::internal::io::StandardErrorOutputStream`:

### Public Member Functions

- **StandardErrorOutputStream** ()
- virtual **~StandardErrorOutputStream** ()
- virtual void **flush** () throw ( `decaf::io::IOException` )

*Flushes this stream by writing any buffered output to the underlying stream.*

#### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

*The default implementation of this method does nothing.*

- virtual void **close** () throw ( `decaf::io::IOException` )

*Closes this object and deallocates the appropriate resources.*

*The object is generally no longer usable after calling close.*

#### Exceptions

<b>IOException</b> (p. 2103)	<i>if an error occurs while closing.</i>
------------------------------	--

*The default implementation of this method does nothing.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

### 6.766.1 Detailed Description

Wrapper Around the Standard error Output facility on the current platform.

This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

### 6.766.2 Constructor & Destructor Documentation

6.766.2.1 decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ( )

6.766.2.2 virtual decaf::internal::io::StandardErrorOutputStream::~~StandardErrorOutputStream ( ) [virtual]

### 6.766.3 Member Function Documentation

6.766.3.1 virtual void decaf::internal::io::StandardErrorOutputStream::close ( ) throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

## Exceptions

<b>IOException</b> (p. 2103)	if an error occurs while closing.
---------------------------------	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2858).

6.766.3.2 virtual void decaf::internal::io::StandardErrorOutputStream::doWriteArrayBounded ( const unsigned char \* buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
[protected, virtual]

Reimplemented from **decaf::io::OutputStream** (p. 2859).

6.766.3.3 virtual void decaf::internal::io::StandardErrorOutputStream::doWriteByte ( unsigned char *value* ) throw ( decaf::io::IOException ) [protected, virtual]

Implements **decaf::io::OutputStream** (p. 2859).

6.766.3.4 virtual void decaf::internal::io::StandardErrorOutputStream::flush ( ) throw ( decaf::io::IOException ) [virtual]

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2859).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardErrorOutputStream.h**

## 6.767 decaf::internal::io::StandardInputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardInputStream.h>
```

Inheritance diagram for decaf::internal::io::StandardInputStream:

### Public Member Functions

- **StandardInputStream** ()
- virtual **~StandardInputStream** ()
- virtual int **available** () const throw ( decaf::io::IOException )

*Indicates the number of bytes available.*

### Protected Member Functions

- virtual int **doReadByte** () throw ( decaf::io::IOException )

### 6.767.1 Constructor & Destructor Documentation

6.767.1.1 `decaf::internal::io::StandardInputStream::StandardInputStream ( )`

6.767.1.2 `virtual decaf::internal::io::StandardInputStream::~~StandardInputStream ( )`  
[virtual]

### 6.767.2 Member Function Documentation

6.767.2.1 `virtual int decaf::internal::io::StandardInputStream::available ( ) const throw ( decaf::io::IOException )` [virtual]

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Reimplemented from **decaf::io::InputStream** (p. 2004).

6.767.2.2 `virtual int decaf::internal::io::StandardInputStream::doReadByte ( ) throw ( decaf::io::IOException )` [protected, virtual]

Implements **decaf::io::InputStream** (p. 2005).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/io/StandardInputStream.h`

## 6.768 decaf::internal::io::StandardOutputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardOutputStream.h>
```

Inheritance diagram for `decaf::internal::io::StandardOutputStream`:

## Public Member Functions

- **StandardOutputStream** ()
- virtual **~StandardOutputStream** ()
- virtual void **flush** () throw ( decaf::io::IOException )

*Flushes this stream by writing any buffered output to the underlying stream.*

### Exceptions

<b>IOException</b> (p. 2103)	<i>if an I/O error occurs.</i>
------------------------------	--------------------------------

*The default implementation of this method does nothing.*

- virtual void **close** () throw ( decaf::io::IOException )

*Closes this object and deallocates the appropriate resources.*

*The object is generally no longer usable after calling close.*

### Exceptions

<b>IOException</b> (p. 2103)	<i>if an error occurs while closing.</i>
------------------------------	--

*The default implementation of this method does nothing.*

## Protected Member Functions

- virtual void **doWriteByte** (unsigned char value) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

## 6.768.1 Constructor & Destructor Documentation

6.768.1.1 decaf::internal::io::StandardOutputStream::StandardOutputStream ( )

6.768.1.2 virtual decaf::internal::io::StandardOutputStream::~~StandardOutputStream ( )  
[virtual]

## 6.768.2 Member Function Documentation

6.768.2.1 virtual void decaf::internal::io::StandardOutputStream::close ( ) throw ( decaf::io::IOException ) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

### Exceptions

<b>IOException</b> (p. 2103)	<i>if an error occurs while closing.</i>
---------------------------------	--

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2858).

```
6.768.2.2  virtual void decaf::internal::io::StandardOutputStream::doWriteArrayBounded
           ( const unsigned char * buffer, int size, int offset, int length ) throw (
           decaf::io::IOException, decaf::lang::exceptions::NullPointerException,
           decaf::lang::exceptions::IndexOutOfBoundsException )
           [protected, virtual]
```

Reimplemented from **decaf::io::OutputStream** (p. 2859).

```
6.768.2.3  virtual void decaf::internal::io::StandardOutputStream::doWriteByte ( unsigned char
           value ) throw ( decaf::io::IOException ) [protected, virtual]
```

Implements **decaf::io::OutputStream** (p. 2859).

```
6.768.2.4  virtual void decaf::internal::io::StandardOutputStream::flush ( ) throw (
           decaf::io::IOException ) [virtual]
```

Flushes this stream by writing any buffered output to the underlying stream.

### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2859).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

## 6.769 cms::Startable Class Reference

Interface for a class that implements the start method.

```
#include <src/main/cms/Startable.h>
```

Inheritance diagram for cms::Startable:



## Public Member Functions

- virtual **~Startable** ()
- virtual void **start** ()=0 throw ( CMSEException )

*Starts the service.*

### 6.769.1 Detailed Description

Interface for a class that implements the start method.

An object that implements the **Startable** (p. 3527) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

#### Since

1.0

### 6.769.2 Constructor & Destructor Documentation

6.769.2.1 virtual cms::Startable::~~Startable ( ) [inline, virtual]

### 6.769.3 Member Function Documentation

6.769.3.1 virtual void cms::Startable::start ( ) throw ( CMSEException ) [pure virtual]

Starts the service.

#### Exceptions

<b>CMSEException</b> (p. 1130)	if an internal error occurs while starting.
-----------------------------------	---

Implemented in **activemq::core::ActiveMQConnection** (p. 263).

The documentation for this class was generated from the following file:

- src/main/cms/**Startable.h**

## 6.770 decaf::lang::STATIC\_CAST\_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

## 6.771 activemq::core::ActiveMQConstants::StaticInitializer Class Reference

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

### Public Member Functions

- **StaticInitializer** ()
- virtual **~StaticInitializer** ()

### Static Public Attributes

- static std::string **destOptions** [NUM\_OPTIONS]
- static std::string **uriParams** [NUM\_PARAMS]
- static std::map< std::string, **DestinationOption** > **destOptionMap**
- static std::map< std::string, **URIParam** > **uriParamsMap**

### 6.771.1 Constructor & Destructor Documentation

6.771.1.1 **activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer** ( )

6.771.1.2 **virtual activemq::core::ActiveMQConstants::StaticInitializer::~~StaticInitializer** ( )  
[inline, virtual]

### 6.771.2 Field Documentation

6.771.2.1 **std::map<std::string, DestinationOption>**  
**activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap**  
[static]

6.771.2.2 **std::string activemq::core::ActiveMQConstants::StaticInitializer::destOptions**[NUM\_OPTIONS]  
[static]

6.771.2.3 **std::string activemq::core::ActiveMQConstants::StaticInitializer::uriParams**[NUM\_PARAMS]  
[static]

6.771.2.4 **std::map<std::string, URIParam>** **activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap**  
[static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConstants.h**

## 6.772 decaf::util::StlList< E > Class Template Reference

**List** (p. 2296) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

```
#include <src/main/decaf/util/StlList.h>
```

Inheritance diagram for decaf::util::StlList< E >:

### Data Structures

- class **ConstStlListIterator**
- class **StlListIterator**

### Public Member Functions

- **StlList** ()  
*Default constructor - does nothing.*
- **StlList** (const **StlList** &source)  
*Copy constructor - copies the content of the given set into this one.*
- **StlList** (const **Collection**< E > &source)  
*Copy constructor - copies the content of the given set into this one.*
- virtual ~**StlList** ()
- virtual bool **equals** (const **StlList** &source) const
- virtual **Iterator**< E > \* **iterator** ()  
**Returns**  
*an iterator over a set of elements of type T.*
- virtual **Iterator**< E > \* **iterator** () const
- virtual **ListIterator**< E > \* **listIterator** ()  
**Returns**  
*a list iterator over the elements in this list (in proper sequence).*
- virtual **ListIterator**< E > \* **listIterator** () const
- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

#### Parameters

index	index of first element to be returned from the list iterator (by a call to the next method).
-------	--

#### Returns

*a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.*

**Exceptions**

IndexOutOfBoundsException	if the index is out of range ( $index < 0 \parallel index > \mathbf{size}()$ (p. 1164))
---------------------------	---

- virtual **ListIterator**< E > \* **listIterator** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual void **copy** (const **StlList** &source)

- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )

*Removes all of the elements from this collection (optional operation).*

*The collection will be empty after this method returns.*

*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2115) operation. Most implementations will probably choose to override this method for efficiency.*

*Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

**Exceptions**

UnsupportedOperationException	if the clear operation is not supported by this collection
-------------------------------	--

- virtual bool **contains** (const E &value) const throw ( lang::Exception )

*Returns true if this collection contains the specified element.*

*This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.*

**Parameters**

value	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1155).
-------	--

**Returns**

*true if the value is contained in this collection*

**Exceptions**

Exception	if an error occurs,
-----------	---------------------

- virtual std::size\_t **indexOf** (const E &value) throw ( decaf::lang::exceptions::NoSuchElementException )

*Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.*

*More formally, returns the lowest index  $i$  such that  $get(i) == value$ , or -1 if there is no such index.*

**Parameters**

value	- element to search for
-------	-------------------------

**Returns**

*the index of the first occurrence of the specified element in this list,*

**Exceptions**

NoSuchElementException	if value is not in the list
------------------------	-----------------------------

- virtual std::size\_t **lastIndexOf** (const E &value) throw ( decaf::lang::exceptions::NoSuchElementException )

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

#### Parameters

value	- element to search for
-------	-------------------------

#### Returns

the index of the last occurrence of the specified element in this list.

#### Exceptions

NoSuchElementException	if value is not in the list
------------------------	-----------------------------

- virtual bool **isEmpty** () const

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 1164) == 0.

#### Returns

true if the size method return 0.

- virtual std::size\_t **size** () const

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

#### Returns

the number of elements in this collection

- virtual E **get** (std::size\_t index) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

Gets the element contained at position passed.

#### Parameters

index	- position to get
-------	-------------------

#### Returns

value at index

- virtual E **set** (std::size\_t index, const E &element) throw ( decaf::lang::exceptions::IndexOutOfBoundsException )

Replaces the element at the specified position in this list with the specified element.

#### Parameters

index	- index of the element to replace
element	- element to be stored at the specified position

#### Returns

the element previously at the specified position

#### Exceptions

IndexOutOfBoundsException	- if the index is greater than size
---------------------------	-------------------------------------

- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1155) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

#### Parameters

value	- reference to the element to add.
-------	------------------------------------

#### Returns

true if the element was added

#### Exceptions

UnsupportedOperationException	
IllegalArgumentException	
IllegalStateException	if the element cannot be added at this time due to insertion restrictions

- virtual void **add** (std::size\_t index, const E &element) throw ( lang::exceptions::UnsupportedOperationException lang::exceptions::IndexOutOfBoundsException )

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

#### Parameters

index	- index at which the specified element is to be inserted
element	- element to be inserted

#### Exceptions

IndexOutOfBoundsException	- if the index is greater than size
UnsupportedOperationException	- If the collection is non-modifiable.

- virtual bool **addAll** (std::size\_t index, const **Collection**< E > &source) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

**Parameters**

index	<i>The index at which to insert the first element from the specified collection</i>
source	<i>The <b>Collection</b> (p. 1155) containing elements to be added to this list</i>

**Returns**

*true if this list changed as a result of the call*

**Exceptions**

IndexOutOfBoundsException	- if the index is greater than size
UnsupportedOperationException	- If the collection is non-modifiable.

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

*More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

*This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.*

*Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.*

**Parameters**

value	- element to be removed from this collection, if present
-------	--

**Returns**

*true if an element was removed as a result of this call*

**Exceptions**

UnsupportedOperationException	<i>if the remove operation is not supported by this collection.</i>
IllegalArgumentException	<i>If the value is not a valid entry for this <b>Collection</b> (p. 1155).</i>

- virtual E **remove** (std::size\_t index) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException )

*Removes the element at the specified position in this list.*

*Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.*

**Parameters**

index	- the index of the element to be removed
-------	--

**Returns**

*the element previously at the specified position*

**Exceptions**

IndexOutOfBoundsException	- if the index is greater than size
UnsupportedOperationException	- If the collection is non-modifiable.

### 6.772.1 Detailed Description

```
template<typename E>class decaf::util::StlList< E >
```

**List** (p. 2296) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

### 6.772.2 Constructor & Destructor Documentation

```
6.772.2.1  template<typename E> decaf::util::StlList< E >::StlList ( ) [inline]
```

Default constructor - does nothing.

```
6.772.2.2  template<typename E> decaf::util::StlList< E >::StlList ( const StlList< E >
            & source ) [inline]
```

Copy constructor - copies the content of the given set into this one.

#### Parameters

<i>source</i>	The source set.
---------------	-----------------

```
6.772.2.3  template<typename E> decaf::util::StlList< E >::StlList ( const Collection<
            E > & source ) [inline]
```

Copy constructor - copies the content of the given set into this one.

#### Parameters

<i>source</i>	The source set.
---------------	-----------------

```
6.772.2.4  template<typename E> virtual decaf::util::StlList< E >::~~StlList ( )
            [inline, virtual]
```

### 6.772.3 Member Function Documentation

```
6.772.3.1  template<typename E> virtual bool decaf::util::StlList< E >::add ( const E
            & value ) throw ( lang::exceptions::UnsupportedOperationException,
            lang::exceptions::IllegalArgumentException,
            lang::exceptions::IllegalStateException ) [inline, virtual]
```

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)



Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1155) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

### Parameters

<i>value</i>	- reference to the element to add.
--------------	------------------------------------

### Returns

true if the element was added

### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 1156).

Referenced by decaf::util::StlList< cms::Connection \* >::addAll().

```
6.772.3.2  template<typename E> virtual void decaf::util::StlList<
            E >::add ( std::size_t index, const E & element ) throw (
            lang::exceptions::UnsupportedOperationException,
            lang::exceptions::IndexOutOfBoundsException ) [inline,
            virtual]
```

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

### Parameters

<i>index</i>	- index at which the specified element is to be inserted
<i>element</i>	- element to be inserted

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2297).

```
6.772.3.3  template<typename E> virtual bool decaf::util::StlList< E >::addAll
( std::size_t index, const Collection< E > & source ) throw (
  decaf::lang::exceptions::UnsupportedOperationException,
  decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
virtual]
```

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

**Parameters**

<i>index</i>	The index at which to insert the first element from the specified collection
<i>source</i>	The <b>Collection</b> (p. 1155) containing elements to be added to this list

**Returns**

true if this list changed as a result of the call

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2298).

```
6.772.3.4  template<typename E> virtual void decaf::util::StlList< E >::clear ( ) throw
( lang::exceptions::UnsupportedOperationException ) [inline,
virtual]
```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iter-**

**ator.remove** (p. 2115) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

### Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 151).

**6.772.3.5** `template<typename E> virtual bool decaf::util::StlList< E >::contains ( const E & value ) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

### Parameters

<i>value</i>	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1155).
--------------	--

### Returns

true if the value is contained in this collection

### Exceptions

<i>Exception</i>	if an error occurs,
------------------	---------------------

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 152).

**6.772.3.6** `template<typename E> virtual void decaf::util::StlList< E >::copy ( const StlList< E > & source ) [inline, virtual]`

Referenced by `decaf::util::StlList< cms::Connection * >::StlList()`.

**6.772.3.7** `template<typename E> virtual bool decaf::util::StlList< E >::equals ( const StlList< E > & source ) const [inline, virtual]`

**6.772.3.8** `template<typename E> virtual E decaf::util::StlList< E >::get ( std::size_t index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Gets the element contained at position passed.

**Parameters**

<i>index</i>	- position to get
--------------	-------------------

**Returns**

value at index

Implements **decaf::util::List**< **E** > (p. 2299).

```
6.772.3.9  template<typename E> virtual std::size_t decaf::util::StlList< E >::indexOf (
            const E & value ) throw ( decaf::lang::exceptions::NoSuchElementException
            ) [inline, virtual]
```

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

**Parameters**

<i>value</i>	- element to search for
--------------	-------------------------

**Returns**

the index of the first occurrence of the specified element in this list,

**Exceptions**

<i>NoSuchElementException</i>	if value is not in the list
-------------------------------	-----------------------------

Implements **decaf::util::List**< **E** > (p. 2299).

```
6.772.3.10  template<typename E> virtual bool decaf::util::StlList< E >::isEmpty ( )
            const [inline, virtual]
```

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 1164) == 0.

**Returns**

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 154).

6.772.3.11 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator ( ) [inline, virtual]`

#### Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p.2113).

6.772.3.12 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator ( ) const [inline, virtual]`

Implements **decaf::lang::Iterable< E >** (p.2114).

6.772.3.13 `template<typename E> virtual std::size_t decaf::util::StlList< E >::lastIndexOf ( const E & value ) throw ( decaf::lang::exceptions::NoSuchElementException ) [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that get(i) == value or -1 if there is no such index.

#### Parameters

<i>value</i>	- element to search for
--------------	-------------------------

#### Returns

the index of the last occurrence of the specified element in this list.

#### Exceptions

<i>NoSuchElementException</i>	if value is not in the list
-------------------------------	-----------------------------

Implements **decaf::util::List< E >** (p.2300).

6.772.3.14 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator ( std::size_t index ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

#### Parameters

<i>index</i>	index of first element to be returned from the list iterator (by a call to the next method).
--------------	--

**Returns**

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

**Exceptions**

<i>IndexOutOfBoundsException</i>	if the index is out of range ( <code>index &lt; 0    index &gt; size()</code> ) (p. 1164))
----------------------------------	--

Implements **decaf::util::List< E >** (p. 2301).

**6.772.3.15** `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator ( ) [inline, virtual]`

**Returns**

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 2300).

**6.772.3.16** `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator ( ) const [inline, virtual]`

Implements **decaf::util::List< E >** (p. 2301).

**6.772.3.17** `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator ( std::size_t index ) const throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Implements **decaf::util::List< E >** (p. 2301).

**6.772.3.18** `template<typename E> virtual E decaf::util::StlList< E >::remove ( std::size_t index ) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException ) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

**Parameters**

<i>index</i>	- the index of the element to be removed
--------------	--

**Returns**

the element previously at the specified position

**Exceptions**

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
<i>UnsupportedOperationException</i>	- If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 2302).

```
6.772.3.19  template<typename E> virtual bool decaf::util::StlList< E >::remove ( const E
            & value ) throw ( lang::exceptions::UnsupportedOperationException,
            lang::exceptions::IllegalArgumentException ) [inline,
            virtual]
```

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

**Parameters**

<i>value</i>	- element to be removed from this collection, if present
--------------	--

**Returns**

true if an element was removed as a result of this call

**Exceptions**

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1155).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 156).

```
6.772.3.20  template<typename E> virtual E decaf::util::StlList<
              E >::set ( std::size_t index, const E & element ) throw (
              decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
              virtual]
```

Replaces the element at the specified position in this list with the specified element.

#### Parameters

<i>index</i>	- index of the element to replace
<i>element</i>	- element to be stored at the specified position

#### Returns

the element previously at the specified position

#### Exceptions

<i>IndexOutOfBoundsException</i>	- if the index is greater than size
----------------------------------	-------------------------------------

Implements **decaf::util::List**< **E** > (p. 2302).

```
6.772.3.21  template<typename E> virtual std::size_t decaf::util::StlList< E >::size ( )
              const [inline, virtual]
```

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

#### Returns

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 1164).

Referenced by **decaf::util::StlList**< **cms::Connection** \* >::add(), **decaf::util::StlList**< **cms::Connection** \* >::addAll(), **decaf::util::StlList**< **cms::Connection** \* >::get(), **decaf::util::StlList**< **cms::Connection** \* >::lastIndexOf(), **decaf::util::StlList**< **cms::Connection** \* >::listIterator(), **decaf::util::StlList**< **cms::Connection** \* >::remove(), and **decaf::util::StlList**< **cms::Connection** \* >::set().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlList.h**

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

**Map** (p. 2419) template that wraps around a **std::map** to provide a more user-friendly interface and to provide common functions that do not exist in **std::map**.



## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 3555

```
#include <src/main/decaf/util/StlMap.h>
```

Inheritance diagram for decaf::util::StlMap< K, V, COMPARATOR >:

### Public Member Functions

- **StlMap** ()  
*Default constructor - does nothing.*
- **StlMap** (const **StlMap** &source)  
*Copy constructor - copies the content of the given map into this one.*
- **StlMap** (const **Map**< K, V, COMPARATOR > &source)  
*Copy constructor - copies the content of the given map into this one.*
- virtual ~**StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const  
*Comparison, equality is dependent on the method of determining if the element are equal.*

#### Parameters

source	- <b>Map</b> (p. 2419) to compare to this one.
--------	--

#### Returns

*true if the **Map** (p. 2419) passed is equal in value to this one.*

- virtual void **copy** (const **StlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)  
*Copies the content of the source map into this map.  
Erases all existing data in this map.*

#### Parameters

source	The source object to copy from.
--------	---------------------------------

- virtual void **clear** () throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*Removes all keys and values from this map.*

#### Exceptions

UnsupportedOperationException	if this map is unmodifiable.
-------------------------------	------------------------------

- virtual bool **containsKey** (const K &key) const  
*Indicates whether or this map contains a value for the given key.*

#### Parameters

key	The key to look up.
-----	---------------------

**Returns**

*true if this map contains the value, otherwise false.*

- virtual bool **containsValue** (const V &value) const

*Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.*

**Parameters**

value	<i>The Value to look up.</i>
-------	------------------------------

**Returns**

*true if this map contains the value, otherwise false.*

- virtual bool **isEmpty** () const

**Returns**

*if the **Map** (p. 2419) contains any element or not, TRUE or FALSE*

- virtual std::size\_t **size** () const

**Returns**

*The number of elements (key/value pairs) in this map.*

- virtual V & **get** (const K &key) throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2419).*

*If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.*

**Parameters**

key	<i>The search key.</i>
-----	------------------------

**Returns**

*A reference to the value for the given key.*

**Exceptions**

NoSuchElementException	<i>if the key requests doesn't exist in the <b>Map</b> (p. 2419).</i>
------------------------	---

- virtual const V & **get** (const K &key) const throw ( lang::exceptions::NoSuchElementException )

*Gets the value mapped to the specified key in the **Map** (p. 2419).*

*If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.*

**Parameters**

key	<i>The search key.</i>
-----	------------------------

**Returns**

*A {const} reference to the value for the given key.*

**Exceptions**

NoSuchElementException	<i>if the key requests doesn't exist in the <b>Map</b> (p. 2419).</i>
------------------------	---

- virtual void **put** (const K &key, const V &value) throw ( decaf::lang::exceptions::UnsupportedOperationException )

*Sets the value for the specified key.*

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 3557

### Parameters

key	The target key.
value	The value to be set.

### Exceptions

UnsupportedOperation Exception	if this map is unmodifiable.
-----------------------------------	------------------------------

- virtual void **putAll** (const **StlMap**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )
- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw ( decaf::lang::exceptions::UnsupportedOperationException )

Stores a copy of the Mappings contained in the other **Map** (p. 2419) in this one.

### Parameters

other	A <b>Map</b> (p. 2419) instance whose elements are to all be inserted in this <b>Map</b> (p. 2419).
-------	---

### Exceptions

UnsupportedOperation Exception	If the implementing class does not support the putAll operation.
-----------------------------------	--

- virtual V **remove** (const K &key) throw ( decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException )

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

### Parameters

key	The search key.
-----	-----------------

### Returns

a copy of the element that was previously mapped to the given key

### Exceptions

NoSuchElementExcep- tion	if this key is not in the <b>Map</b> (p. 2419).
UnsupportedOperation Exception	if this map is unmodifiable.

- virtual std::vector< K > **keySet** () const

Returns a **Set** (p. 3379) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2115), **Set.remove** (p. 156), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

### Returns

the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

**Returns**

*the entire set of values in this map as a std::vector.*

- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )

*Locks the object.*

- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )

*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*

- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )

*Unlocks the object.*

- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

**6.773.1 Detailed Description**

```
template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::StlMap<
K, V, COMPARATOR >
```

**Map** (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

**Since**

1.0

**6.773.2 Constructor & Destructor Documentation**

```
6.773.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>>
decaf::util::StlMap< K, V, COMPARATOR >::StlMap ( ) [inline]
```

Default constructor - does nothing.

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 3559

6.773.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
decaf::util::StlMap< K, V, COMPARATOR >::StlMap ( const StlMap< K, V,  
COMPARATOR > & source ) [inline]`

Copy constructor - copies the content of the given map into this one.

### Parameters

<i>source</i>	The source map.
---------------	-----------------

6.773.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
decaf::util::StlMap< K, V, COMPARATOR >::StlMap ( const Map< K, V,  
COMPARATOR > & source ) [inline]`

Copy constructor - copies the content of the given map into this one.

### Parameters

<i>source</i>	The source map.
---------------	-----------------

6.773.2.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual decaf::util::StlMap< K, V, COMPARATOR >::~~StlMap ( ) [inline,  
virtual]`

## 6.773.3 Member Function Documentation

6.773.3.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::StlMap< K, V, COMPARATOR >::clear ( ) throw  
( decaf::lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Removes all keys and values from this map.

### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2421).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

6.773.3.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsKey ( const K &  
key ) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

**Parameters**

<i>key</i>	The key to look up.
------------	---------------------

**Returns**

true if this map contains the value, otherwise false.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2421).

Referenced by decaf::util::StlMap< std::string, cms::Topic \* >::equals().

```
6.773.3.3  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsValue ( const V
           & value ) const [inline, virtual]
```

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

**Parameters**

<i>value</i>	The Value to look up.
--------------	-----------------------

**Returns**

true if this map contains the value, otherwise false.

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2422).

```
6.773.3.4  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy ( const StlMap<
           K, V, COMPARATOR > & source ) [inline, virtual]
```

Referenced by decaf::util::StlMap< std::string, cms::Topic \* >::StlMap().

```
6.773.3.5  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy ( const Map< K, V,
           COMPARATOR > & source ) [inline, virtual]
```

Copies the content of the source map into this map.

Erases all existing data in this map.

**Parameters**

<i>source</i>	The source object to copy from.
---------------	---------------------------------

Implements **decaf::util::Map**< **K**, **V**, **COMPARATOR** > (p. 2423).

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 3561

6.773.3.6 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals ( const Map< K,  
V, COMPARATOR > & source ) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

### Parameters

<i>source</i>	- <b>Map</b> (p. 2419) to compare to this one.
---------------	--

### Returns

true if the **Map** (p. 2419) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2423).

6.773.3.7 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals ( const StlMap<  
K, V, COMPARATOR > & source ) const [inline, virtual]`

6.773.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual V& decaf::util::StlMap< K, V, COMPARATOR >::get ( const K & key )  
throw ( lang::exceptions::NoSuchElementException ) [inline,  
virtual]`

Gets the value mapped to the specified key in the **Map** (p. 2419).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

### Parameters

<i>key</i>	The search key.
------------	-----------------

### Returns

A reference to the value for the given key.

### Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2419).
-------------------------------	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2424).

```
6.773.3.9  template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual const V& decaf::util::StlMap< K, V, COMPARATOR >::get ( const K & key )
           const throw ( lang::exceptions::NoSuchElementException ) [inline,
           virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 2419).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

#### Parameters

<i>key</i>	The search key.
------------	-----------------

#### Returns

A {const} reference to the value for the given key.

#### Exceptions

<i>NoSuchElementException</i>	if the key requests doesn't exist in the <b>Map</b> (p. 2419).
-------------------------------	--

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2425).

```
6.773.3.10 template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual bool decaf::util::StlMap< K, V, COMPARATOR >::isEmpty ( ) const
           [inline, virtual]
```

#### Returns

if the **Map** (p. 2419) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2426).

```
6.773.3.11 template<typename K, typename V, typename COMPARATOR = std::less<K>>
           virtual std::vector<K> decaf::util::StlMap< K, V, COMPARATOR >::keySet ( )
           const [inline, virtual]
```

Returns a **Set** (p. 3379) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the `setValue` operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 2115), **Set.remove** (p. 156), `removeAll`, `retainAll` and `clear` operations. It does not support the `add` or `addAll` operations.

#### Returns

the entire set of keys in this map as a `std::vector`.



### 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 3563

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2426).

```
6.773.3.12  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::lock ( ) throw (
            decaf::lang::exceptions::RuntimeException ) [inline, virtual]
```

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

```
6.773.3.13  template<typename K, typename V, typename COMPARATOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
            >::notify ( ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
            virtual]
```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

```
6.773.3.14  template<typename K, typename V, typename COMPARATOR = std::less<K>>
            virtual void decaf::util::StlMap< K, V, COMPARATOR >::notifyAll
            ( ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
            virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

```
6.773.3.15  template<typename K, typename V, typename COMPARTOR
            = std::less<K>> virtual void decaf::util::StlMap< K, V,
            COMPARTOR >::put ( const K & key, const V & value ) throw (
            decaf::lang::exceptions::UnsupportedOperationException )
            [inline, virtual]
```

Sets the value for the specified key.

#### Parameters

<i>key</i>	The target key.
<i>value</i>	The value to be set.

#### Exceptions

<i>UnsupportedOperationException</i>	if this map is unmodifiable.
--------------------------------------	------------------------------

Implements **decaf::util::Map< K, V, COMPARTOR >** (p. 2427).

Referenced by **decaf::util::StlMap< std::string, cms::Topic \* >::putAll()**.

```
6.773.3.16  template<typename K, typename V, typename COMPARTOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARTOR
            >::putAll ( const StlMap< K, V, COMPARTOR > & other ) throw (
            decaf::lang::exceptions::UnsupportedOperationException )
            [inline, virtual]
```

Referenced by **decaf::util::StlMap< std::string, cms::Topic \* >::copy()**.

```
6.773.3.17  template<typename K, typename V, typename COMPARTOR =
            std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARTOR
            >::putAll ( const Map< K, V, COMPARTOR > & other ) throw (
            decaf::lang::exceptions::UnsupportedOperationException )
            [inline, virtual]
```

Stores a copy of the Mappings contained in the other **Map** (p. 2419) in this one.

#### Parameters

<i>other</i>	A <b>Map</b> (p. 2419) instance whose elements are to all be inserted in this <b>Map</b> (p. 2419).
--------------	---

#### Exceptions

<i>UnsupportedOperationException</i>	If the implementing class does not support the putAll operation.
--------------------------------------	--

## 6.773 `decaf::util::StlMap< K, V, COMPARATOR >` Class Template Reference 3565

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2428).

```
6.773.3.18  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove ( const K
             & key ) throw ( decaf::lang::exceptions::NoSuchElementException,
             decaf::lang::exceptions::UnsupportedOperationException )
             [inline, virtual]
```

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

### Parameters

<i>key</i>	The search key.
------------	-----------------

### Returns

a copy of the element that was previously mapped to the given key

### Exceptions

<i>NoSuchElementException</i>	if this key is not in the <b>Map</b> (p. 2419).
<i>UnsupportedOperationException</i>	if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2429).

```
6.773.3.19  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual std::size_t decaf::util::StlMap< K, V, COMPARATOR >::size ( ) const
             [inline, virtual]
```

### Returns

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 2430).

```
6.773.3.20  template<typename K, typename V, typename COMPARATOR = std::less<K>>
             virtual bool decaf::util::StlMap< K, V, COMPARATOR >::tryLock ( ) throw (
             decaf::lang::exceptions::RuntimeException ) [inline, virtual]
```

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

```
6.773.3.21  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual void decaf::util::StlMap< K, V, COMPARATOR >::unlock ( ) throw (
decaf::lang::exceptions::RuntimeException ) [inline, virtual]
```

Unlocks the object.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

```
6.773.3.22  template<typename K, typename V, typename COMPARATOR = std::less<K>>
virtual std::vector<V> decaf::util::StlMap< K, V, COMPARATOR >::values ( )
const [inline, virtual]
```

**Returns**

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 2430).

Referenced by decaf::util::StlMap< std::string, cms::Topic \* >::values().

```
6.773.3.23  template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::wait ( ) throw ( decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [inline,
virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

## 6.773 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference 3567

6.773.3.24 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( long long  
    millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException,  
    decaf::lang::exceptions::IllegalArgumentException,  
    decaf::lang::exceptions::IllegalMonitorStateException,  
    decaf::lang::exceptions::InterruptedException ) [inline,  
virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

### Exceptions

<i>IllegalArgumentEx- ception</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorState- Exception</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

6.773.3.25 `template<typename K, typename V, typename COMPARATOR = std::less<K>>  
virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait ( long long  
    millisecs ) throw ( decaf::lang::exceptions::RuntimeException,  
    decaf::lang::exceptions::IllegalMonitorStateException,  
    decaf::lang::exceptions::InterruptedException ) [inline,  
virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

**Exceptions**

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlMap.h**

**6.774 decaf::util::StlQueue< T > Class Template Reference**

The **Queue** (p. 3094) class accepts messages with an psuh(m) command where m is the message to be queued.

```
#include <src/main/decaf/util/StlQueue.h>
```

Inheritance diagram for decaf::util::StlQueue< T >:

**Data Structures**

- class **QueueIterator**

**Public Member Functions**

- **StlQueue** ()
- virtual **~StlQueue** ()
- **Iterator**< T > \* **iterator** ()  
*Gets an **Iterator** (p. 2114) over this **Queue** (p. 3094).*
- void **clear** ()  
*Empties this queue.*
- T & **front** ()  
*Returns a Reference to the element at the head of the queue.*
- const T & **front** () const  
*Returns a Reference to the element at the head of the queue.*
- T & **back** ()  
*Returns a Reference to the element at the tail of the queue.*
- const T & **back** () const  
*Returns a Reference to the element at the tail of the queue.*
- void **push** (const T &t)  
*Places a new Object at the Tail of the queue.*

- void **enqueueFront** (const T &t)  
*Places a new Object at the front of the queue.*
- T **pop** ()  
*Removes and returns the element that is at the Head of the queue.*
- size\_t **size** () const  
*Gets the Number of elements currently in the **Queue** (p. 3094).*
- bool **empty** () const  
*Checks if this **Queue** (p. 3094) is currently empty.*
- virtual std::vector< T > **toArray** () const
- void **reverse** (StlQueue< T > &target) const  
*Reverses the order of the contents of this queue and stores them in the target queue.*
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*
- T & **getSafeValue** ()  
*Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.*

### 6.774.1 Detailed Description

```
template<typename T>class decaf::util::StlQueue< T >
```

The **Queue** (p. 3094) class accepts messages with an psuh(m) command where m is the message to be queued.

It destructively returns the message with **pop()** (p. 3561). **pop()** (p. 3561) returns messages in the order they were enqueued.

**Queue** (p. 3094) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the pop method actually reaturns a reference to the element popped. This frees the app from having to call the **front** method before calling pop.

```
Queue<string> sq; // make a queue to hold string messages sq.push(s); // enqueues
a message m string s = sq.pop(); // dequeues a message
```

#### = DESIGN CONSIDERATIONS

The **Queue** (p. 3094) class inherits from the Synchronizable interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p. 3094).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.

## 6.774.2 Constructor & Destructor Documentation

```
6.774.2.1 template<typename T> decaf::util::StlQueue< T >::StlQueue ( )
[inline]
```

```
6.774.2.2 template<typename T> virtual decaf::util::StlQueue< T >::~~StlQueue ( )
[inline, virtual]
```

## 6.774.3 Member Function Documentation

```
6.774.3.1 template<typename T> T& decaf::util::StlQueue< T >::back ( ) [inline]
```

Returns a Reference to the element at the tail of the queue.

### Returns

reference to a queue type object or (safe)

```
6.774.3.2 template<typename T> const T& decaf::util::StlQueue< T >::back ( ) const
[inline]
```

Returns a Reference to the element at the tail of the queue.

### Returns

reference to a queue type object or (safe)



6.774.3.3 `template<typename T> void decaf::util::StlQueue< T >::clear ( )`  
`[inline]`

Empties this queue.

6.774.3.4 `template<typename T> bool decaf::util::StlQueue< T >::empty ( ) const`  
`[inline]`

Checks if this **Queue** (p. 3094) is currently empty.

#### Returns

boolean indicating queue emptiness

6.774.3.5 `template<typename T> void decaf::util::StlQueue< T >::enqueueFront ( const`  
`T & t ) [inline]`

Places a new Object at the front of the queue.

#### Parameters

<code>t</code> - <b>Queue</b> (p. 3094) Object Type reference.
--

6.774.3.6 `template<typename T> const T& decaf::util::StlQueue< T >::front ( ) const`  
`[inline]`

Returns a Reference to the element at the head of the queue.

#### Returns

reference to a queue type object or (safe)

6.774.3.7 `template<typename T> T& decaf::util::StlQueue< T >::front ( ) [inline]`

Returns a Reference to the element at the head of the queue.

#### Returns

reference to a queue type object or (safe)

6.774.3.8 `template<typename T> T& decaf::util::StlQueue< T >::getSafeValue ( )`  
`[inline]`

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

**Returns**

Reference to this Queues safe object

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::back()`, `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::front()`, and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::pop()`.

**6.774.3.9** `template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator ( )`  
`[inline]`

Gets an **Iterator** (p. 2114) over this **Queue** (p. 3094).

**Returns**

new iterator pointer that is owned by the caller.

**6.774.3.10** `template<typename T> virtual void decaf::util::StlQueue< T >::lock ( )`  
`throw ( decaf::lang::exceptions::RuntimeException ) [inline,`  
`virtual]`

Locks the object.

**Exceptions**

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

**6.774.3.11** `template<typename T> virtual void decaf::util::StlQueue< T >::notify`  
`( ) throw ( decaf::lang::exceptions::RuntimeException,`  
`decaf::lang::exceptions::IllegalMonitorStateException ) [inline,`  
`virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

**Exceptions**

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

```
6.774.3.12  template<typename T> virtual void decaf::util::StlQueue< T >::notifyAll
            ( ) throw ( decaf::lang::exceptions::RuntimeException,
                        decaf::lang::exceptions::IllegalMonitorStateException ) [inline,
virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

```
6.774.3.13  template<typename T> T decaf::util::StlQueue< T >::pop ( ) [inline]
```

Removes and returns the element that is at the Head of the queue.

### Returns

reference to a queue type object or (safe)

```
6.774.3.14  template<typename T> void decaf::util::StlQueue< T >::push ( const T & t )
            [inline]
```

Places a new Object at the Tail of the queue.

### Parameters

<i>t</i>	- <b>Queue</b> (p. 3094) Object Type reference.
----------	---

```
6.774.3.15  template<typename T> void decaf::util::StlQueue< T >::reverse (
            StlQueue< T > & target ) const [inline]
```

Reverses the order of the contents of this queue and stores them in the target queue.

### Parameters

<i>target</i>	- The target queue that will receive the contents of this queue in reverse order.
---------------	---

6.774.3.16 `template<typename T> size_t decaf::util::StlQueue< T >::size ( ) const`  
`[inline]`

Gets the Number of elements currently in the **Queue** (p. 3094).

#### Returns

**Queue** (p. 3094) Size

6.774.3.17 `template<typename T> virtual std::vector<T> decaf::util::StlQueue< T`  
`>::toArray ( ) const [inline, virtual]`

#### Returns

the all values in this queue as a std::vector.

6.774.3.18 `template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock (`  
`) throw ( decaf::lang::exceptions::RuntimeException ) [inline,`  
`virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

#### Returns

true if the lock was acquired, false if it is already held by another thread.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

6.774.3.19 `template<typename T> virtual void decaf::util::StlQueue< T >::unlock (`  
`) throw ( decaf::lang::exceptions::RuntimeException ) [inline,`  
`virtual]`

Unlocks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

```
6.774.3.20  template<typename T> virtual void decaf::util::StlQueue< T >::wait
            ( ) throw ( decaf::lang::exceptions::RuntimeException,
                        decaf::lang::exceptions::IllegalMonitorStateException,
                        decaf::lang::exceptions::InterruptedException ) [inline,
                                virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

```
6.774.3.21  template<typename T> virtual void decaf::util::StlQueue< T >::wait ( long
            long millisecs ) throw ( decaf::lang::exceptions::RuntimeException,
                        decaf::lang::exceptions::IllegalMonitorStateException,
                        decaf::lang::exceptions::InterruptedException ) [inline,
                                virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

```
6.774.3.22  template<typename T> virtual void decaf::util::StlQueue< T >::wait ( long long
            millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalArgumentException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline,
            virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p.3653).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlQueue.h**

## 6.775 decaf::util::StlSet< E > Class Template Reference

**Set** (p.3379) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

```
#include <src/main/decaf/util/StlSet.h>
```

Inheritance diagram for decaf::util::StlSet< E >:

## Data Structures

- class **ConstSetIterator**
- class **SetIterator**

## Public Member Functions

- **StlSet** ()  
*Default constructor - does nothing.*
- **StlSet** (const **StlSet** &source)  
*Copy constructor - copies the content of the given set into this one.*
- **StlSet** (const **Collection**< E > &source)  
*Copy constructor - copies the content of the given set into this one.*
- virtual ~**StlSet** ()
- **Iterator**< E > \* **iterator** ()  
**Returns**  
*an iterator over a set of elements of type T.*
- **Iterator**< E > \* **iterator** () const
- virtual bool **equals** (const **StlSet** &source) const
- virtual void **copy** (const **StlSet** &source)
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all of the elements from this collection (optional operation).  
The collection will be empty after this method returns.  
This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 2115) operation. Most implementations will probably choose to override this method for efficiency.  
Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*

### Exceptions

UnsupportedOperationException	if the clear operation is not supported by this collection
-------------------------------	--

- virtual bool **contains** (const E &value) const throw ( lang::Exception )  
*Returns true if this collection contains the specified element.  
This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.*

### Parameters

value	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1155).
-------	--

### Returns

*true if the value is contained in this collection*

### Exceptions

Exception	if an error occurs,
-----------	---------------------

- virtual bool **isEmpty** () const
- virtual std::size\_t **size** () const
- virtual bool **add** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException )

*Returns true if this collection changed as a result of the call.*

*(Returns false if this collection does not permit duplicates and already contains the specified element.)*

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1155) classes should clearly specify in their documentation any restrictions on what elements may be added.*

*If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.*

*For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.*

#### Parameters

value	- reference to the element to add.
-------	------------------------------------

#### Returns

*true if the element was added*

#### Exceptions

UnsupportedOperationException	
IllegalArgumentException	
IllegalStateException	<i>if the element cannot be added at this time due to insertion restrictions</i>

- virtual bool **remove** (const E &value) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

*Removes a single instance of the specified element from this collection, if it is present (optional operation).*

*More formally, removes the first element *e* such that *e == o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

*This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.*

*Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.*

#### Parameters

value	- element to be removed from this collection, if present
-------	--

#### Returns

*true if an element was removed as a result of this call*



**Exceptions**

UnsupportedOperationException	<i>if the remove operation is not supported by this collection.</i>
IllegalArgumentException	<i>If the value is not a valid entry for this <b>Collection</b> (p. 1155).</i>

**6.775.1 Detailed Description**

```
template<typename E> class decaf::util::StlSet< E >
```

**Set** (p. 3379) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

**6.775.2 Constructor & Destructor Documentation**

```
6.775.2.1 template<typename E> decaf::util::StlSet< E >::StlSet ( ) [inline]
```

Default constructor - does nothing.

```
6.775.2.2 template<typename E> decaf::util::StlSet< E >::StlSet ( const StlSet< E >
& source ) [inline]
```

Copy constructor - copies the content of the given set into this one.

**Parameters**

<i>source</i>	The source set.
---------------	-----------------

```
6.775.2.3 template<typename E> decaf::util::StlSet< E >::StlSet ( const Collection<
E > & source ) [inline]
```

Copy constructor - copies the content of the given set into this one.

**Parameters**

<i>source</i>	The source set.
---------------	-----------------

```
6.775.2.4 template<typename E> virtual decaf::util::StlSet< E >::~~StlSet ( )
[inline, virtual]
```

**6.775.3 Member Function Documentation**

```
6.775.3.1  template<typename E> virtual bool decaf::util::StlSet< E >::add ( const E
           & value ) throw ( lang::exceptions::UnsupportedOperationException,
           lang::exceptions::IllegalArgumentException,
           lang::exceptions::IllegalStateException ) [inline, virtual]
```

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 1155) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

#### Parameters

<i>value</i>	- reference to the element to add.
--------------	------------------------------------

#### Returns

true if the element was added

#### Exceptions

<i>UnsupportedOperationException</i>	
<i>IllegalArgumentException</i>	
<i>IllegalStateException</i>	if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 1156).

```
6.775.3.2  template<typename E> virtual void decaf::util::StlSet< E >::clear ( ) throw
           ( lang::exceptions::UnsupportedOperationException ) [inline,
           virtual]
```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iter-**

**ator.remove** (p. 2115) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

### Exceptions

<i>UnsupportedOperationException</i>	if the clear operation is not supported by this collection
--------------------------------------	--

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 151).

**6.775.3.3** `template<typename E> virtual bool decaf::util::StlSet< E >::contains ( const E & value ) const throw ( lang::Exception ) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

### Parameters

<i>value</i>	- the value whose presence is to be queried for in this <b>Collection</b> (p. 1155).
--------------	--

### Returns

true if the value is contained in this collection

### Exceptions

<i>Exception</i>	if an error occurs,
------------------	---------------------

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 152).

**6.775.3.4** `template<typename E> virtual void decaf::util::StlSet< E >::copy ( const StlSet< E > & source ) [inline, virtual]`

Referenced by `decaf::util::StlSet< ActiveMQSession * >::StlSet()`.

**6.775.3.5** `template<typename E> virtual bool decaf::util::StlSet< E >::equals ( const StlSet< E > & source ) const [inline, virtual]`

**6.775.3.6** `template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty ( ) const [inline, virtual]`

### Returns

if the set contains any element or not, TRUE or FALSE

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 154).

```
6.775.3.7  template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ( )
           const [inline, virtual]
```

Implements **decaf::lang::Iterable**< **E** > (p. 2114).

```
6.775.3.8  template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator ( )
           [inline, virtual]
```

### Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable**< **E** > (p. 2113).

```
6.775.3.9  template<typename E> virtual bool decaf::util::StlSet< E >::remove ( const E
           & value ) throw ( lang::exceptions::UnsupportedOperationException,
           lang::exceptions::IllegalArgumentException ) [inline, virtual]
```

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

### Parameters

<i>value</i>	- element to be removed from this collection, if present
--------------	--

### Returns

true if an element was removed as a result of this call

### Exceptions

<i>UnsupportedOperationException</i>	if the remove operation is not supported by this collection.
<i>IllegalArgumentException</i>	If the value is not a valid entry for this <b>Collection</b> (p. 1155).

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 156).

## 6.776 `activemq::wireformat::stomp::StompCommandConstants` Class Reference

6.775.3.10 `template<typename E> virtual std::size_t decaf::util::StlSet< E >::size ( )`  
`const [inline, virtual]`

### Returns

The number of elements in this set.

Implements `decaf::util::Collection< E >` (p. 1164).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlSet.h`

## 6.776 `activemq::wireformat::stomp::StompCommandConstants` Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompCommandConstants.h>
```

### Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**
- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR\_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER\_DESTINATION**
- static const std::string **HEADER\_TRANSACTIONID**
- static const std::string **HEADER\_CONTENTLENGTH**
- static const std::string **HEADER\_SESSIONID**
- static const std::string **HEADER\_RECEIPT\_REQUIRED**
- static const std::string **HEADER\_RECEIPTID**
- static const std::string **HEADER\_MESSAGEID**
- static const std::string **HEADER\_ACK**
- static const std::string **HEADER\_LOGIN**
- static const std::string **HEADER\_PASSWORD**
- static const std::string **HEADER\_CLIENT\_ID**
- static const std::string **HEADER\_MESSAGE**
- static const std::string **HEADER\_CORRELATIONID**

- static const std::string **HEADER\_REQUESTID**
- static const std::string **HEADER\_RESPONSEID**
- static const std::string **HEADER\_EXPIRES**
- static const std::string **HEADER\_PERSISTENT**
- static const std::string **HEADER\_REPLYTO**
- static const std::string **HEADER\_TYPE**
- static const std::string **HEADER\_DISPATCH\_ASYNC**
- static const std::string **HEADER\_EXCLUSIVE**
- static const std::string **HEADER\_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER\_NOLOCAL**
- static const std::string **HEADER\_PREFETCHSIZE**
- static const std::string **HEADER\_JMSPRIORITY**
- static const std::string **HEADER\_CONSUMERPRIORITY**
- static const std::string **HEADER\_RETROACTIVE**
- static const std::string **HEADER\_SUBSCRIPTIONNAME**
- static const std::string **HEADER\_OLDSUBSCRIPTIONNAME**
- static const std::string **HEADER\_TIMESTAMP**
- static const std::string **HEADER\_REDELIVERED**
- static const std::string **HEADER\_REDELIVERYCOUNT**
- static const std::string **HEADER\_SELECTOR**
- static const std::string **HEADER\_ID**
- static const std::string **HEADER\_SUBSCRIPTION**
- static const std::string **HEADER\_TRANSFORMATION**
- static const std::string **HEADER\_TRANSFORMATION\_ERROR**
- static const std::string **ACK\_CLIENT**
- static const std::string **ACK\_AUTO**
- static const std::string **ACK\_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE\_PREFIX**
- static const std::string **TOPIC\_PREFIX**
- static const std::string **TEMPQUEUE\_PREFIX**
- static const std::string **TEMPTOPIC\_PREFIX**

### 6.776.1 Field Documentation

- 6.776.1.1 `const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT`  
[static]
- 6.776.1.2 `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK`  
[static]
- 6.776.1.3 `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_ -`  
`AUTO` [static]

## 6.776 `activemq::wireformat::stomp::StompCommandConstants` Class Reference

- 6.776.1.4 `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_CLIENT` [static]
- 6.776.1.5 `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_INDIVIDUAL` [static]
- 6.776.1.6 `const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN` [static]
- 6.776.1.7 `const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES` [static]
- 6.776.1.8 `const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT` [static]
- 6.776.1.9 `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT` [static]
- 6.776.1.10 `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED` [static]
- 6.776.1.11 `const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT` [static]
- 6.776.1.12 `const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD` [static]
- 6.776.1.13 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK` [static]
- 6.776.1.14 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID` [static]
- 6.776.1.15 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY` [static]
- 6.776.1.16 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONTENTLENGTH` [static]
- 6.776.1.17 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CORRELATIONID` [static]
- 6.776.1.18 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_DESTINATION` [static]
- 6.776.1.19 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_DISPATCH_ASYNC` [static]

- 6.776.1.20 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-EXCLUSIVE [static]`
- 6.776.1.21 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-EXPIRES [static]`
- 6.776.1.22 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-ID [static]`
- 6.776.1.23 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-JMSPRIORITY [static]`
- 6.776.1.24 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-LOGIN [static]`
- 6.776.1.25 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-MAXPENDINGMSGLIMIT [static]`
- 6.776.1.26 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-MESSAGE [static]`
- 6.776.1.27 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-MESSAGEID [static]`
- 6.776.1.28 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-NOLOCAL [static]`
- 6.776.1.29 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-OLDSUBSCRIPTIONNAME [static]`
- 6.776.1.30 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-PASSWORD [static]`
- 6.776.1.31 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-PERSISTENT [static]`
- 6.776.1.32 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-PREFETCHSIZE [static]`
- 6.776.1.33 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-RECEIPT_REQUIRED [static]`
- 6.776.1.34 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-RECEIPTID [static]`
- 6.776.1.35 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_-REDELIVERED [static]`



## 6.776 `activemq::wireformat::stomp::StompCommandConstants` Class Reference

- 6.776.1.36 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REDELIVERYCOUNT` [static]
- 6.776.1.37 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REPLYTO` [static]
- 6.776.1.38 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_REQUESTID` [static]
- 6.776.1.39 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RESPONSEID` [static]
- 6.776.1.40 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_RETROACTIVE` [static]
- 6.776.1.41 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SELECTOR` [static]
- 6.776.1.42 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SESSIONID` [static]
- 6.776.1.43 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SUBSCRIPTION` [static]
- 6.776.1.44 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_SUBSCRIPTIONNAME` [static]
- 6.776.1.45 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TIMESTAMP` [static]
- 6.776.1.46 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSACTIONID` [static]
- 6.776.1.47 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSFORMATION` [static]
- 6.776.1.48 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TRANSFORMATION_ERROR` [static]
- 6.776.1.49 `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_TYPE` [static]
- 6.776.1.50 `const std::string activemq::wireformat::stomp::StompCommandConstants::MESSAGE` [static]
- 6.776.1.51 `const std::string activemq::wireformat::stomp::StompCommandConstants::QUEUE_PREFIX` [static]

- 6.776.1.52 `const std::string activemq::wireformat::stomp::StompCommandConstants::RECEIPT`  
[static]
- 6.776.1.53 `const std::string activemq::wireformat::stomp::StompCommandConstants::SEND`  
[static]
- 6.776.1.54 `const std::string activemq::wireformat::stomp::StompCommandConstants::SUBSCRIBE`  
[static]
- 6.776.1.55 `const std::string activemq::wireformat::stomp::StompCommandConstants::TEMPQUEUE_  
PREFIX` [static]
- 6.776.1.56 `const std::string activemq::wireformat::stomp::StompCommandConstants::TEMPTOPIC_  
PREFIX` [static]
- 6.776.1.57 `const std::string activemq::wireformat::stomp::StompCommandConstants::TEXT`  
[static]
- 6.776.1.58 `const std::string activemq::wireformat::stomp::StompCommandConstants::TOPIC_  
PREFIX` [static]
- 6.776.1.59 `const std::string activemq::wireformat::stomp::StompCommandConstants::UNSUBSCRIBE`  
[static]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompCommandConstants.h`

## 6.777 activemq::wireformat::stomp::StompFrame Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.

```
#include <src/main/activemq/wireformat/stomp/StompFrame.h>
```

### Public Member Functions

- **StompFrame** ()  
*Default constructor.*
- virtual **~StompFrame** ()  
*Destruction.*
- **StompFrame \* clone** () const  
*Clone this message exactly, returns a new instance that the caller is required to delete.*
- void **copy** (const **StompFrame** \*src)  
*Copies the contents of the passed Frame to this one.*
- void **setCommand** (const std::string &cmd)

*Sets the command for this stomp frame.*

- const std::string & **getCommand** () const

*Accessor for this frame's command field.*

- bool **hasProperty** (const std::string &name) const

*Checks if the given property is present in the Frame.*

- std::string **getProperty** (const std::string &name, const std::string &fallback="") const

*Gets a property from this Frame's properties and returns it, or the default value given.*

- std::string **removeProperty** (const std::string &name)

*Gets and remove the property specified, if the property is not set, this method returns the empty string.*

- void **setProperty** (const std::string &name, const std::string &value)

*Sets the property given to the value specified in this Frame's Properties.*

- **decaf::util::Properties** & **getProperties** ()

*Gets access to the header properties for this frame.*

- const **decaf::util::Properties** & **getProperties** () const

- const std::vector< unsigned char > & **getBody** () const

*Accessor for the body data of this frame.*

- std::vector< unsigned char > & **getBody** ()

*Non-const version of the body accessor.*

- std::size\_t **getBodyLength** () const

*Return the number of bytes contained in this frames body.*

- void **setBody** (const unsigned char \*bytes, std::size\_t numBytes)

*Sets the body data of this frame as a byte sequence.*

- void **toStream** (**decaf::io::DataOutputStream** \*stream) const throw ( decaf::io::IOException )

*Writes this Frame to an OuputStream in the Stomp Wire Format.*

- void **fromStream** (**decaf::io::DataInputStream** \*stream) throw ( decaf::io::IOException )

*Reads a Stop Frame from a DataInputStream in the Stomp Wire format.*

### 6.777.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

### 6.777.2 Constructor & Destructor Documentation

#### 6.777.2.1 activemq::wireformat::stomp::StompFrame::StompFrame ( ) [inline]

Default constructor.

6.777.2.2 `virtual activemq::wireformat::stomp::StompFrame::~~StompFrame ( ) [inline, virtual]`

Destruction.

### 6.777.3 Member Function Documentation

6.777.3.1 `StompFrame* activemq::wireformat::stomp::StompFrame::clone ( ) const`

Clone this message exactly, returns a new instance that the caller is required to delete.

#### Returns

new copy of this message

6.777.3.2 `void activemq::wireformat::stomp::StompFrame::copy ( const StompFrame * src )`

Copies the contents of the passed Frame to this one.

#### Parameters

<i>src</i>	- Frame to copy
------------	-----------------

6.777.3.3 `void activemq::wireformat::stomp::StompFrame::fromStream ( decaf::io::DataInputStream * stream ) throw ( decaf::io::IOException )`

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

#### Parameters

<i>stream</i>	- The stream to read the Frame from.
---------------	--------------------------------------

#### Exceptions

<i>IOException</i>	if an error occurs while writing the Frame.
--------------------	---

6.777.3.4 `const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody ( ) const [inline]`

Accessor for the body data of this frame.

#### Returns

char pointer to body data

6.777.3.5 `std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody ( ) [inline]`

Non-const version of the body accessor.

6.777.3.6 `std::size_t activemq::wireformat::stomp::StompFrame::getBodyLength ( ) const [inline]`

Return the number of bytes contained in this frames body.

#### Returns

Body bytes length.

6.777.3.7 `const std::string& activemq::wireformat::stomp::StompFrame::getCommand ( ) const [inline]`

Accessor for this frame's command field.

6.777.3.8 `decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ( ) [inline]`

Gets access to the header properties for this frame.

#### Returns

the Properties object owned by this Frame

6.777.3.9 `const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties ( ) const [inline]`

6.777.3.10 `std::string activemq::wireformat::stomp::StompFrame::getProperty ( const std::string & name, const std::string & fallback = "" ) const [inline]`

Gets a property from this Frame's properties and returns it, or the default value given.

#### Parameters

<i>name</i>	- The name of the property to lookup
<i>fallback</i>	- The default value to return if this value isn't set

#### Returns

string value of the property asked for.

6.777.3.11 `bool activemq::wireformat::stomp::StompFrame::hasProperty ( const std::string & name ) const` `[inline]`

Checks if the given property is present in the Frame.

#### Parameters

<i>name</i>	- The name of the property to check for.
-------------	--

6.777.3.12 `std::string activemq::wireformat::stomp::StompFrame::removeProperty ( const std::string & name )` `[inline]`

Gets and remove the property specified, if the property is not set, this method returns the empty string.

#### Parameters

<i>name</i>	- the Name of the property to get and return.
-------------	---

6.777.3.13 `void activemq::wireformat::stomp::StompFrame::setBody ( const unsigned char * bytes, std::size_t numBytes )`

Sets the body data of this frame as a byte sequence.

#### Parameters

<i>bytes</i>	The byte buffer to be set in the body.
<i>numBytes</i>	The number of bytes in the buffer.

6.777.3.14 `void activemq::wireformat::stomp::StompFrame::setCommand ( const std::string & cmd )` `[inline]`

Sets the command for this stomp frame.

#### Parameters

<i>cmd</i>	command The command to be set.
------------	--------------------------------

6.777.3.15 `void activemq::wireformat::stomp::StompFrame::setProperty ( const std::string & name, const std::string & value )` `[inline]`

Sets the property given to the value specified in this Frame's Properties.

#### Parameters

<i>name</i>	- Name of the property.
<i>value</i>	- Value to set the property to.

6.777.3.16 void activemq::wireformat::stomp::StompFrame::toStream ( decaf::io::DataOutputStream \* *stream* ) const throw ( decaf::io::IOException )

Writes this Frame to an OuputStream in the Stomp Wire Format.

#### Parameters

<i>stream</i>	- The stream to write the Frame to.
---------------	-------------------------------------

#### Exceptions

<i>IOException</i>	if an error occurs while reading the Frame.
--------------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompFrame.h**

## 6.778 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from StompFrame's.

```
#include <src/main/activemq/wireformat/stomp/StompHelper.h>
```

#### Public Member Functions

- **StompHelper** ()
- virtual ~**StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)  
*Converts the Headers in a Stomp Frame into Headers in the given Message Command.*
- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)  
*Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3576).*
- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)  
*Converts from a Stomp Destination to an ActiveMQDestination.*
- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)

*Converts from a ActiveMQDestination to a Stomp Destination Name.*

- `std::string convertMessageld (const Pointer< Messageld > &messageld)`

*Converts a Messageld instance to a Stomp Messageld String.*

- `Pointer< Messageld > convertMessageld (const std::string &messageld)`

*Converts a Stomp Messageld string to a Messageld.*

- `std::string convertConsumerId (const Pointer< ConsumerId > &consumerId)`

*Converts a ConsumerId instance to a Stomp ConsumerId String.*

- `Pointer< ConsumerId > convertConsumerId (const std::string &consumerId)`

*Converts a Stomp ConsumerId string to a ConsumerId.*

- `std::string convertProducerId (const Pointer< ProducerId > &producerId)`

*Converts a ProducerId instance to a Stomp ProducerId String.*

- `Pointer< ProducerId > convertProducerId (const std::string &producerId)`

*Converts a Stomp ProducerId string to a ProducerId.*

- `std::string convertTransactionId (const Pointer< TransactionId > &transactionId)`

*Converts a TransactionId instance to a Stomp TransactionId String.*

- `Pointer< TransactionId > convertTransactionId (const std::string &transactionId)`

*Converts a Stomp TransactionId string to a TransactionId.*

### 6.778.1 Detailed Description

Utility Methods used when marshaling to and from StompFrame's.

Since

3.0

### 6.778.2 Constructor & Destructor Documentation

6.778.2.1 `activemq::wireformat::stomp::StompHelper::StompHelper ( ) [inline]`

6.778.2.2 `virtual activemq::wireformat::stomp::StompHelper::~~StompHelper ( ) [inline, virtual]`

### 6.778.3 Member Function Documentation

6.778.3.1 `std::string activemq::wireformat::stomp::StompHelper::convertConsumerId ( const Pointer< ConsumerId > & consumerId )`

Converts a ConsumerId instance to a Stomp ConsumerId String.

Parameters



<i>consumerId</i>	- the Consumer instance to convert.
-------------------	-------------------------------------

**Returns**

a Stomp Consumer Id String.

**6.778.3.2** `Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId ( const std::string & consumerId )`

Converts a Stomp ConsumerId string to a ConsumerId.

**Parameters**

<i>consumerId</i>	- the String Consumer Id to convert.
-------------------	--------------------------------------

**Returns**

Pointer to a new ConsumerId.

**6.778.3.3** `std::string activemq::wireformat::stomp::StompHelper::convertDestination ( const Pointer< ActiveMQDestination > & destination )`

Converts from a ActiveMQDestination to a Stomp Destination Name.

**Parameters**

<i>destination</i>	- The ActiveMQDestination to Convert
--------------------	--------------------------------------

**Returns**

the Stomp String name that defines the destination.

**6.778.3.4** `Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination ( const std::string & destination )`

Converts from a Stomp Destination to an ActiveMQDestination.

**Parameters**

<i>destination</i>	- The Stomp Destination name string.
--------------------	--------------------------------------

**Returns**

Pointer to a new ActiveMQDestination.

6.778.3.5 `std::string activemq::wireformat::stomp::StompHelper::convertMessageld ( const Pointer< Messageld > & messageld )`

Converts a Messageld instance to a Stomp Messageld String.

#### Parameters

<i>messageld</i>	- the Messageld instance to convert.
------------------	--------------------------------------

#### Returns

a Stomp Message Id String.

6.778.3.6 `Pointer<Messageld> activemq::wireformat::stomp::StompHelper::convertMessageld ( const std::string & messageld )`

Converts a Stomp Messageld string to a Messageld.

#### Parameters

<i>messageld</i>	- the String message Id to convert.
------------------	-------------------------------------

#### Returns

Pointer to a new Messageld.

6.778.3.7 `std::string activemq::wireformat::stomp::StompHelper::convertProducerId ( const Pointer< ProducerId > & producerId )`

Converts a ProducerId instance to a Stomp ProducerId String.

#### Parameters

<i>producerId</i>	- the Producer instance to convert.
-------------------	-------------------------------------

#### Returns

a Stomp Producer Id String.

6.778.3.8 `Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId ( const std::string & producerId )`

Converts a Stomp ProducerId string to a ProducerId.

#### Parameters

<i>producerId</i>	- the String Producer Id to convert.
-------------------	--------------------------------------

**Returns**

Pointer to a new ProducerId.

**6.778.3.9** void activemq::wireformat::stomp::StompHelper::convertProperties ( const  
Pointer< Message > & message, const Pointer< StompFrame > & frame )

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 3576).

**Parameters**

<i>message</i>	- The message to move the Headers to.
<i>frame</i>	- The frame to extract headers from.

**6.778.3.10** void activemq::wireformat::stomp::StompHelper::convertProperties ( const  
Pointer< StompFrame > & frame, const Pointer< Message > & message )

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

**Parameters**

<i>frame</i>	- The frame to extract headers from.
<i>message</i>	- The message to move the Headers to.

**6.778.3.11** Pointer<TransactionId> activemq::wireformat::stomp::StompHelper::convertTransactionId  
( const std::string & transactionId )

Converts a Stomp TransactionId string to a TransactionId.

**Parameters**

<i>transactionId</i>	- the String Transaction Id to convert.
----------------------	---

**Returns**

Pointer to a new TransactionId.

**6.778.3.12** std::string activemq::wireformat::stomp::StompHelper::convertTransactionId ( const  
Pointer< TransactionId > & transactionId )

Converts a TransactionId instance to a Stomp TransactionId String.

**Parameters**

<i>transactionId</i>	- the Transaction instance to convert.
----------------------	--

**Returns**

a Stomp Transaction Id String.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompHelper.h**

## 6.779 **activemq::wireformat::stomp::StompWireFormat** Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompWireFormat.h>
```

Inheritance diagram for **activemq::wireformat::stomp::StompWireFormat**:

**Public Member Functions**

- **StompWireFormat** ()
- virtual **~StompWireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out) throw ( **decaf::io::IOException** )  
*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in) throw ( **decaf::io::IOException** )  
*Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.*
- virtual void **setVersion** (int version **AMQCPP\_UNUSED**)  
*Set the Version.*
- virtual int **getVersion** () const  
*Get the Version.*
- virtual bool **inReceive** () const  
*Is there a Message being unmarshaled?*
- virtual bool **hasNegotiator** () const  
*Returns true if this **WireFormat** (p. 3907) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport) throw ( **decaf::lang::exceptions::UnsupportedOperationException** )  
*If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.*

## 6.779.1 Constructor & Destructor Documentation

6.779.1.1 `activemq::wireformat::stomp::StompWireFormat::StompWireFormat ( )`

6.779.1.2 `virtual activemq::wireformat::stomp::StompWireFormat::~~StompWireFormat ( )`  
[virtual]

## 6.779.2 Member Function Documentation

6.779.2.1 `virtual Pointer<transport::Transport> activemq::wireformat::stomp::StompWireFormat::createNegotiator ( const Pointer< transport::Transport > & transport ) throw ( decaf::lang::exceptions::UnsupportedOperationException )`  
[virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

### Returns

new instance of a **WireFormatNegotiator** (p. 3946).

### Exceptions

<i>UnsupportedOperationException</i>	if the <b>WireFormat</b> (p. 3907) doesn't have a Negotiator.
--------------------------------------	---

Implements **activemq::wireformat::WireFormat** (p. 3908).

6.779.2.2 `virtual int activemq::wireformat::stomp::StompWireFormat::getVersion ( ) const`  
[inline, virtual]

Get the Version.

### Returns

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3909).

6.779.2.3 `virtual bool activemq::wireformat::stomp::StompWireFormat::hasNegotiator ( ) const`  
[inline, virtual]

Returns true if this **WireFormat** (p. 3907) has a Negotiator that needs to wrap the Transport that uses it.

### Returns

true if the **WireFormat** (p. 3907) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3909).

6.779.2.4 `virtual bool activemq::wireformat::stomp::StompWireFormat::inReceive ( ) const`  
`[inline, virtual]`

Is there a Message being unmarshaled?

### Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3909).

6.779.2.5 `virtual void activemq::wireformat::stomp::StompWireFormat::marshal`  
`( const Pointer< commands::Command > & command, const`  
`activemq::transport::Transport * transport, decaf::io::DataOutputStream`  
`* out ) throw ( decaf::io::IOException ) [virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

### Parameters

<i>command</i>	The Command to Marshal to the output stream.
<i>transport</i>	The Transport that initiated this marshal call.
<i>out</i>	The output stream to write the command to.

### Exceptions

<i>IOException</i>	
--------------------	--

Implements **activemq::wireformat::WireFormat** (p. 3910).

6.779.2.6 `virtual void activemq::wireformat::stomp::StompWireFormat::setVersion ( int version`  
`AMQCPP_UNUSED ) [inline, virtual]`

Set the Version.

### Parameters

<i>the</i>	version of the wire format
------------	----------------------------

Implements **activemq::wireformat::WireFormat** (p. 3910).

## 6.780 `activemq::wireformat::stomp::StompWireFormatFactory` Class Reference 1601

6.779.2.7 `virtual Pointer<commands::Command>`  
`activemq::wireformat::stomp::StompWireFormat::unmarshal ( const`  
`activemq::transport::Transport * transport, decaf::io::DataInputStream *`  
`in ) throw ( decaf::io::IOException ) [virtual]`

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

### Parameters

<i>transport</i>	- Pointer to the transport that is making this request.
<i>in</i>	- the input stream to read the command from.

### Returns

the newly marshaled Command, caller owns the pointer

### Exceptions

<i>IOException</i>	
--------------------	--

Implements `activemq::wireformat::WireFormat` (p. 3910).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormat.h`

## 6.780 `activemq::wireformat::stomp::StompWireFormatFactory` Class Reference

Factory used to create the Stomp Wire Format instance.

```
#include <src/main/activemq/wireformat/stomp/StompWireFormatFactory.h>
```

Inheritance diagram for `activemq::wireformat::stomp::StompWireFormatFactory`:

### Public Member Functions

- `StompWireFormatFactory ()`
- `virtual ~StompWireFormatFactory ()`
- `virtual Pointer< WireFormat > createWireFormat (const decaf::util::Properties &properties) throw ( decaf::lang::exceptions::IllegalStateException )`  
*Creates a new **WireFormat** (p. 3907) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.780.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

### 6.780.2 Constructor & Destructor Documentation

6.780.2.1 `activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory ( )` `[inline]`

6.780.2.2 `virtual activemq::wireformat::stomp::StompWireFormatFactory::~~StompWireFormatFactory ( )` `[inline, virtual]`

### 6.780.3 Member Function Documentation

6.780.3.1 `virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat ( const decaf::util::Properties & properties ) throw ( decaf::lang::exceptions::IllegalStateException )` `[virtual]`

Creates a new **WireFormat** (p. 3907) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters

<i>properties</i>	- the Properties for this <b>WireFormat</b> (p. 3907)
-------------------	---

Implements `activemq::wireformat::WireFormatFactory` (p. 3912).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormatFactory.h`

## 6.781 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

```
#include <src/main/cms/Stoppable.h>
```

Inheritance diagram for cms::Stoppable:

### Public Member Functions

- `virtual ~Stoppable ()`
- `virtual void stop ()=0 throw ( CMSException )`  
*Stops this service.*



### 6.781.1 Detailed Description

Interface for a class that implements the stop method.

An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

#### Since

1.0

### 6.781.2 Constructor & Destructor Documentation

6.781.2.1 `virtual cms::Stoppable::~~Stoppable ( ) [inline, virtual]`

### 6.781.3 Member Function Documentation

6.781.3.1 `virtual void cms::Stoppable::stop ( ) throw ( CMSException ) [pure virtual]`

Stops this service.

#### Exceptions

<b>CMSException</b> (p. 1130)	- if an internal error occurs while stopping the Service.
----------------------------------	---

Implemented in **activemq::core::ActiveMQConnection** (p. 263).

The documentation for this class was generated from the following file:

- `src/main/cms/Stoppable.h`

## 6.782 decaf::util::logging::StreamHandler Class Reference

Stream based logging **Handler** (p. 1941).

```
#include <src/main/decaf/util/logging/StreamHandler.h>
```

Inheritance diagram for decaf::util::logging::StreamHandler:

#### Public Member Functions

- **StreamHandler ( )**

Create a **StreamHandler** (p. 3591), with no current output stream.

- **StreamHandler** (decaf::io::OutputStream \*stream, **Formatter** \*formatter)

Create a **StreamHandler** (p. 3591), with no current output stream.

- virtual ~**StreamHandler** ()
- virtual void **close** () throw ( decaf::io::IOException )

Close the current output stream.

- virtual void **flush** ()

Flush the Handler's output, clears any buffers.

- virtual void **publish** (const **LogRecord** &record)

Publish the Log Record to this **Handler** (p. 1941).

- virtual bool **isLoggable** (const **LogRecord** &record) const

Check if this **Handler** (p. 1941) would actually log a given **LogRecord** (p. 2370).

## Protected Member Functions

- virtual void **setOutputStream** (decaf::io::OutputStream \*stream) throw ( decaf::lang::exceptions::NullPointerException )

Change the output stream.

- void **close** (bool closeStream)

Closes this handler, but the underlying output stream is only closed if closeStream is true.

### 6.782.1 Detailed Description

Stream based logging **Handler** (p. 1941).

This is primarily intended as a base class or support class to be used in implementing other logging Handlers.

LogRecords are published to a given **decaf::io::OutputStream** (p. 2856).

Configuration: By default each **StreamHandler** (p. 3591) is initialized using the following **LogManager** (p. 2363) configuration properties. If properties are not defined (or have invalid values) then the specified default values are used.

\* decaf.util.logging.StreamHandler.level specifies the default level for the **Handler** (p. 1941) (defaults to **Level.INFO** (p. 2295)). \* decaf.util.logging.StreamHandler.filter specifies the name of a **Filter** (p. 1853) class to use (defaults to no **Filter** (p. 1853)). \* decaf.util.logging.StreamHandler.formatter specifies the name of a **Formatter** (p. 1927) class to use (defaults to **decaf.util.logging.SimpleFormatter** (p. 3442)).

Since

1.0

## 6.782.2 Constructor & Destructor Documentation

### 6.782.2.1 decaf::util::logging::StreamHandler::StreamHandler ( )

Create a **StreamHandler** (p. 3591), with no current output stream.

### 6.782.2.2 decaf::util::logging::StreamHandler::StreamHandler ( decaf::io::OutputStream \* *stream*, Formatter \* *formatter* )

Create a **StreamHandler** (p. 3591), with no current output stream.

### 6.782.2.3 virtual decaf::util::logging::StreamHandler::~StreamHandler ( ) [virtual]

## 6.782.3 Member Function Documentation

### 6.782.3.1 virtual void decaf::util::logging::StreamHandler::close ( ) throw ( decaf::io::IOException ) [virtual]

Close the current output stream.

The close method will perform a flush and then close the **Handler** (p. 1941). After close has been called this **Handler** (p. 1941) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

### Exceptions

<i>IOException</i>	if an I/O error occurs.
--------------------	-------------------------

Implements **decaf::io::Closeable** (p. 1121).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 1368).

### 6.782.3.2 void decaf::util::logging::StreamHandler::close ( bool *closeStream* ) [protected]

Closes this handler, but the underlying output stream is only closed if closeStream is true.

### Parameters

<i>closeStream</i>	whether to close the underlying output stream.
--------------------	--

### 6.782.3.3 virtual void decaf::util::logging::StreamHandler::flush ( ) [virtual]

Flush the Handler's output, clears any buffers.

Implements **decaf::util::logging::Handler** (p. 1942).

6.782.3.4 `virtual bool decaf::util::logging::StreamHandler::isLoggable ( const LogRecord & record ) const` [virtual]

Check if this **Handler** (p. 1941) would actually log a given **LogRecord** (p. 2370).

#### Parameters

<i>record</i>	The <b>LogRecord</b> (p. 2370) to check
---------------	---

#### Returns

true if the record can be logged with current settings.

Reimplemented from **decaf::util::logging::Handler** (p. 1943).

6.782.3.5 `virtual void decaf::util::logging::StreamHandler::publish ( const LogRecord & record )` [virtual]

Publish the Log Record to this **Handler** (p. 1941).

#### Parameters

<i>record</i>	The <b>LogRecord</b> (p. 2370) to Publish
---------------	---

Implements **decaf::util::logging::Handler** (p. 1944).

Reimplemented in **decaf::util::logging::ConsoleHandler** (p. 1368).

6.782.3.6 `virtual void decaf::util::logging::StreamHandler::setOutputStream ( decaf::io::OutputStream * stream ) throw ( decaf::lang::exceptions::NullPointerException )` [protected, virtual]

Change the output stream.

If there is a current output stream then the Formatter's tail string is written and the stream is flushed and closed. Then the output stream is replaced with the new output stream.

#### Parameters

<i>stream</i>	The new output stream. May not be NULL.
---------------	---

#### Exceptions

<i>NullPointerException</i>	if the passed stream is NULL.
-----------------------------	-------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

## 6.783 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 3595).

```
#include <src/main/cms/StreamMessage.h>
```

Inheritance diagram for cms::StreamMessage:

### Public Member Functions

- virtual **~StreamMessage** ()
- virtual bool **readBoolean** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a Boolean from the Stream message stream.*
- virtual void **writeBoolean** (bool value)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a boolean to the Stream message stream as a 1-byte value.*
- virtual unsigned char **readByte** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a Byte from the Stream message stream.*
- virtual void **writeByte** (unsigned char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a byte to the Stream message stream as a 1-byte value.*
- virtual int **readBytes** (std::vector< unsigned char > &value) const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a byte array from the Stream message stream.*
- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.*
- virtual int **readBytes** (unsigned char \*buffer, int length) const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a portion of the Stream message stream.*
- virtual void **writeBytes** (const unsigned char \*value, int offset, int length)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a portion of a byte array to the Stream message stream.*
- virtual char **readChar** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException )  
*Reads a Char from the Stream message stream.*
- virtual void **writeChar** (char value)=0 throw ( cms::MessageNotWriteableException, cms::CMSException )  
*Writes a char to the Stream message stream as a 1-byte value.*

- virtual float **readFloat** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit float from the Stream message stream.*
- virtual void **writeFloat** (float value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a float to the Stream message stream as a 4 byte value.*
- virtual double **readDouble** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit double from the Stream message stream.*
- virtual void **writeDouble** (double value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a double to the Stream message stream as a 8 byte value.*
- virtual short **readShort** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit signed short from the Stream message stream.*
- virtual void **writeShort** (short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed short to the Stream message stream as a 2 byte value.*
- virtual unsigned short **readUnsignedShort** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 16 bit unsigned short from the Stream message stream.*
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a unsigned short to the Stream message stream as a 2 byte value.*
- virtual int **readInt** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 32 bit signed integer from the Stream message stream.*
- virtual void **writeInt** (int value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a signed int to the Stream message stream as a 4 byte value.*
- virtual long long **readLong** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads a 64 bit long from the Stream message stream.*
- virtual void **writeLong** (long long value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes a long long to the Stream message stream as a 8 byte value.*
- virtual std::string **readString** () const =0 throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException )  
*Reads an ASCII String from the Stream message stream.*
- virtual void **writeString** (const std::string &value)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )  
*Writes an ASCII String to the Stream message stream.*

### 6.783.1 Detailed Description

Interface for a **StreamMessage** (p. 3595).

The stream Messages provides a **Message** (p. 2493) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

**StreamMessage** (p. 3595) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 1130). The string-to- primitive conversions may throw a runtime exception if the primitive's valueOf() method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X
double								X	X	X
String	X	X	X		X	X	X	X	X	X
byte[]										X

Since

1.3

### 6.783.2 Constructor & Destructor Documentation

6.783.2.1 virtual cms::StreamMessage::~StreamMessage ( ) [inline, virtual]

### 6.783.3 Member Function Documentation

6.783.3.1 virtual bool cms::StreamMessage::readBoolean ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a Boolean from the Stream message stream.

#### Returns

boolean value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 511).

```
6.783.3.2 virtual unsigned char cms::StreamMessage::readByte ( ) const throw (
    cms::MessageEOFException, cms::MessageFormatException,
    cms::MessageNotReadableException, cms::CMSException ) [pure
    virtual]
```

Reads a Byte from the Stream message stream.

#### Returns

unsigned char value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 511).

```
6.783.3.3 virtual int cms::StreamMessage::readBytes ( std::vector< unsigned
    char > & value ) const throw ( cms::MessageEOFException,
    cms::MessageFormatException, cms::MessageNotReadableException,
    cms::CMSException ) [pure virtual]
```

Reads a byte array from the Stream message stream.



If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

#### Parameters

<i>value</i>	buffer to place data in
--------------	-------------------------

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b>MessageEOFException</b> (p. 2621)	- if unexpected end of message stream has been reached.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.
<b>MessageNotReadableException</b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 513).

```
6.783.3.4 virtual int cms::StreamMessage::readBytes ( unsigned char * buffer, int length ) const
throw ( cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException ) [pure
virtual]
```

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p. 1130) is thrown. No bytes will be read from the stream for this exception

case.

#### Parameters

<i>buffer</i>	the buffer into which the data is read
<i>length</i>	the number of bytes to read; must be less than or equal to value.length

#### Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 512).

```
6.783.3.5 virtual char cms::StreamMessage::readChar ( ) const throw (
    cms::MessageEOFException, cms::MessageFormatException,
    cms::MessageNotReadableException, cms::CMSException ) [pure
    virtual]
```

Reads a Char from the Stream message stream.

#### Returns

char value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 513).

6.783.3.6 virtual double cms::StreamMessage::readDouble ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a 64 bit double from the Stream message stream.

#### Returns

double value from stream

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b>MessageEOFException</b> (p. 2621)	- if unexpected end of message stream has been reached.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.
<b>MessageNotReadableException</b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 514).

6.783.3.7 virtual float cms::StreamMessage::readFloat ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException ) [pure virtual]

Reads a 32 bit float from the Stream message stream.

#### Returns

double value from stream

#### Exceptions

<b>CMSException</b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b>MessageEOFException</b> (p. 2621)	- if unexpected end of message stream has been reached.
<b>MessageFormatException</b> (p. 2622)	- if this type conversion is invalid.

<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.
--	---

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 514).

```
6.783.3.8 virtual int cms::StreamMessage::readInt ( ) const throw (
    cms::MessageEOFException, cms::MessageFormatException,
    cms::MessageNotReadableException, cms::CMSException ) [pure
    virtual]
```

Reads a 32 bit signed integer from the Stream message stream.

#### Returns

int value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 515).

```
6.783.3.9 virtual long long cms::StreamMessage::readLong ( ) const throw (
    cms::MessageEOFException, cms::MessageFormatException,
    cms::MessageNotReadableException, cms::CMSException ) [pure
    virtual]
```

Reads a 64 bit long from the Stream message stream.

#### Returns

long long value from stream

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
---	---

<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 515).

6.783.3.10 virtual short cms::StreamMessage::readShort ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]

Reads a 16 bit signed short from the Stream message stream.

#### Returns

short value from stream

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 516).

6.783.3.11 virtual std::string cms::StreamMessage::readString ( ) const throw ( cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException ) [pure virtual]

Reads an ASCII String from the Stream message stream.

**Returns**

String from stream

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 516).

```
6.783.3.12 virtual unsigned short cms::StreamMessage::readUnsignedShort ( ) const throw
( cms::MessageEOFException, cms::MessageFormatException,
cms::MessageNotReadableException, cms::CMSException ) [pure
virtual]
```

Reads a 16 bit unsigned short from the Stream message stream.

**Returns**

unsigned short value from stream

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to read the message due to some internal error.
<b><i>MessageEOFException</i></b> (p. 2621)	- if unexpected end of message stream has been reached.
<b><i>MessageFormatException</i></b> (p. 2622)	- if this type conversion is invalid.
<b><i>MessageNotReadableException</i></b> (p. 2679)	- if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 517).

6.783.3.13 `virtual void cms::StreamMessage::writeBoolean ( bool value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

#### Parameters

<i>value</i>	boolean to write to the stream
--------------	--------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 518).

6.783.3.14 `virtual void cms::StreamMessage::writeByte ( unsigned char value ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

#### Parameters

<i>value</i>	byte to write to the stream
--------------	-----------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 518).

6.783.3.15 `virtual void cms::StreamMessage::writeBytes ( const unsigned char * value, int offset, int length ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]`

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

**Parameters**

<i>value</i>	bytes to write to the stream
<i>offset</i>	the initial offset within the byte array
<i>length</i>	the number of bytes to use

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 519).

```
6.783.3.16 virtual void cms::StreamMessage::writeBytes ( const std::vector< unsigned
char > & value ) throw ( cms::MessageNotWriteableException,
cms::CMSException ) [pure virtual]
```

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

**Parameters**

<i>value</i>	bytes to write to the stream
--------------	------------------------------

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 519).

```
6.783.3.17 virtual void cms::StreamMessage::writeChar ( char value ) throw (
cms::MessageNotWriteableException, cms::CMSException ) [pure
virtual]
```

Writes a char to the Stream message stream as a 1-byte value.

**Parameters**

<i>value</i>	char to write to the stream
--------------	-----------------------------

**Exceptions**

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
---	--



<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.
---	--

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 520).

6.783.3.18 virtual void cms::StreamMessage::writeDouble ( double *value* ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]

Writes a double to the Stream message stream as a 8 byte value.

#### Parameters

<i>value</i>	double to write to the stream
--------------	-------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 520).

6.783.3.19 virtual void cms::StreamMessage::writeFloat ( float *value* ) throw ( cms::MessageNotWriteableException, cms::CMSException ) [pure virtual]

Writes a float to the Stream message stream as a 4 byte value.

#### Parameters

<i>value</i>	float to write to the stream
--------------	------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 520).

6.783.3.20 `virtual void cms::StreamMessage::writeInt ( int value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

#### Parameters

<i>value</i>	signed int to write to the stream
--------------	-----------------------------------

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 521).

6.783.3.21 `virtual void cms::StreamMessage::writeLong ( long long value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a long long to the Stream message stream as a 8 byte value.

#### Parameters

<i>value</i>	signed long long to write to the stream
--------------	---

#### Exceptions

<b><i>CMSException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in `activemq::commands::ActiveMQStreamMessage` (p. 521).

6.783.3.22 `virtual void cms::StreamMessage::writeShort ( short value ) throw ( cms::MessageNotWriteableException, cms::CMSException )` [pure virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

#### Parameters

<i>value</i>	signed short to write to the stream
--------------	-------------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 522).

6.783.3.23 `virtual void cms::StreamMessage::writeString ( const std::string & value ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Writes an ASCII String to the Stream message stream.

**Parameters**

<i>value</i>	String to write to the stream
--------------	-------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 522).

6.783.3.24 `virtual void cms::StreamMessage::writeUnsignedShort ( unsigned short value ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Writes a unsigned short to the Stream message stream as a 2 byte value.

**Parameters**

<i>value</i>	unsigned short to write to the stream
--------------	---------------------------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if the CMS provider fails to write the message due to some internal error.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 522).

The documentation for this class was generated from the following file:

- src/main/cms/**StreamMessage.h**

## 6.784 decaf::lang::String Class Reference

The **String** (p. 3610) class represents an immutable sequence of chars.

```
#include <src/main/decaf/lang/String.h>
```

Inheritance diagram for decaf::lang::String:

### Public Member Functions

- **String** ()

*Creates a new empty **String** (p. 3610) object.*

- **String** (const std::string &source)

*Create a new **String** (p. 3610) object that represents the given STL string.*

- virtual ~**String** ()
- bool **isEmpty** () const
- virtual int **length** () const

#### Returns

*the length of the underlying character sequence.*

- virtual char **charAt** (int index) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Returns the Char at the specified index so long as the index is not greater than the length of the sequence.*

#### Parameters

index	- position to return the char at.
-------	-----------------------------------

#### Returns

*the char at the given position*

#### Exceptions

IndexOutOfBoundsException	if index is > than <b>length()</b> (p. 1108) or negative
---------------------------	--

- virtual **CharSequence** \* **subSequence** (int start, int end) const throw ( lang::exceptions::IndexOutOfBoundsException )

*Returns a new **CharSequence** (p. 1107) that is a subsequence of this sequence.*

*The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.*

#### Parameters

start	- the start index, inclusive
end	- the end index, exclusive

**Returns**

a new **CharSequence** (p. 1107)

**Exceptions**

IndexOutOfBoundsException	if start or end > <b>length()</b> (p. 1108) or start or end are negative.
---------------------------	---

- virtual std::string **toString** () const

**Returns**

the string representation of this **CharSequence** (p. 1107)

**6.784.1 Detailed Description**

The **String** (p. 3610) class represents an immutable sequence of chars.

**Since**

1.0

**6.784.2 Constructor & Destructor Documentation****6.784.2.1 decaf::lang::String::String ( )**

Creates a new empty **String** (p. 3610) object.

**6.784.2.2 decaf::lang::String::String ( const std::string & source )**

Create a new **String** (p. 3610) object that represents the given STL string.

**Parameters**

<i>source</i>	The string to copy into this new <b>String</b> (p. 3610) object.
---------------	--

**6.784.2.3 virtual decaf::lang::String::~~String ( ) [virtual]****6.784.3 Member Function Documentation****6.784.3.1 virtual char decaf::lang::String::charAt ( int index ) const throw ( lang::exceptions::IndexOutOfBoundsException ) [virtual]**

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

**Parameters**

<i>index</i>	- position to return the char at.
--------------	-----------------------------------

**Returns**

the char at the given position

**Exceptions**

<i>IndexOutOfBoundsException</i>	if index is > than <b>length()</b> (p. 1108) or negative
----------------------------------	--

Implements **decaf::lang::CharSequence** (p. 1108).

6.784.3.2 **bool** decaf::lang::String::isEmpty ( ) **const**

**Returns**

true if the length of this **String** (p. 3610) is zero.

6.784.3.3 **virtual int** decaf::lang::String::length ( ) **const** [virtual]

**Returns**

the length of the underlying character sequence.

Implements **decaf::lang::CharSequence** (p. 1108).

6.784.3.4 **virtual CharSequence\*** decaf::lang::String::subSequence ( int *start*, int *end* ) **const**  
**throw ( lang::exceptions::IndexOutOfBoundsException )** [virtual]

Returns a new **CharSequence** (p. 1107) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

**Parameters**

<i>start</i>	- the start index, inclusive
<i>end</i>	- the end index, exclusive

**Returns**

a new **CharSequence** (p. 1107)

**Exceptions**

<i>IndexOutOfBoundsException</i>	if start or end > <b>length()</b> (p. 1108) or start or end are negative.
----------------------------------	---

Implements **decaf::lang::CharSequence** (p. 1109).

6.784.3.5 virtual std::string decaf::lang::String::toString ( ) const [virtual]

### Returns

the string representation of this **CharSequence** (p. 1107)

Implements **decaf::lang::CharSequence** (p. 1109).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**String.h**

## 6.785 decaf::util::StringTokenizer Class Reference

```
#include <src/main/decaf/util/StringTokenizer.h>
```

### Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)  
*Constructs a string tokenizer for the specified string.*
- virtual ~**StringTokenizer** ()
- virtual int **countTokens** () const  
*Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.*
- virtual bool **hasMoreTokens** () const  
*Tests if there are more tokens available from this tokenizer's string.*
- virtual std::string **nextToken** () throw ( lang::exceptions::NoSuchElementException )  
*Returns the next token from this string tokenizer.*
- virtual std::string **nextToken** (const std::string &delim) throw ( lang::exceptions::NoSuchElementException )  
*Returns the next token in this string tokenizer's string.*
- virtual unsigned int **toArray** (std::vector< std::string > &array)  
*Grab all remaining tokens in the String and return them in the vector that is passed in by reference.*
- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)  
*Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.*

### 6.785.1 Constructor & Destructor Documentation

**6.785.1.1** `decaf::util::StringTokenizer::StringTokenizer ( const std::string & str, const std::string & delim = " \t\n\r\f", bool returnDelims = false )`

Constructs a string tokenizer for the specified string.

All characters in the *delim* argument are the delimiters for separating tokens.

If the *returnDelims* flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if *delim* is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p.3613) may result in an Exception.

#### Parameters

<i>str</i>	- The string to tokenize
<i>delim</i>	- String containing the delimiters
<i>returnDelims</i>	- boolean indicating if the delimiters are returned as tokens

**6.785.1.2** `virtual decaf::util::StringTokenizer::~StringTokenizer ( ) [virtual]`

### 6.785.2 Member Function Documentation

**6.785.2.1** `virtual int decaf::util::StringTokenizer::countTokens ( ) const [virtual]`

Calculates the number of times that this tokenizer's `nextToken` method can be called before it generates an exception.

The current position is not advanced.

#### Returns

Count of remaining tokens

**6.785.2.2** `virtual bool decaf::util::StringTokenizer::hasMoreTokens ( ) const [virtual]`

Tests if there are more tokens available from this tokenizer's string.

#### Returns

true if there are more tokens remaining

**6.785.2.3** `virtual std::string decaf::util::StringTokenizer::nextToken ( const std::string & delim ) throw ( lang::exceptions::NoSuchElementException ) [virtual]`

Returns the next token in this string tokenizer's string.



First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 3613) object is changed to be the characters in the string `delim`. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

**Parameters**

<i>delim</i>	- string containing the new set of delimiters
--------------	---

**Returns**

next string in the token list

**Exceptions**

<i>NoSuchElementException</i>	
-------------------------------	--

```
6.785.2.4 virtual std::string decaf::util::StringTokenizer::nextToken ( ) throw (
    lang::exceptions::NoSuchElementException ) [virtual]
```

Returns the next token from this string tokenizer.

**Returns**

string value of next token

**Exceptions**

<i>NoSuchElementException</i>	
-------------------------------	--

```
6.785.2.5 virtual void decaf::util::StringTokenizer::reset ( const std::string & str = "", const
    std::string & delim = "", bool returnDelims = false ) [virtual]
```

Resets the Tokenizer's position in the String to the Beginning calls to `countToken` and `nextToken` now start back at the beginning.

This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing. If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. If set the delim param will reset the string that this Tokenizer is using to tokenize the string. If set to "", no change is made. If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

**Parameters**

<i>str</i>	- New String to tokenize or "", defaults to ""
<i>delim</i>	- New Delimiter String to use or "", defaults to ""
<i>returnDelims</i>	- Should the Tokenizer return delimiters as Tokens, default false

6.785.2.6 `virtual unsigned int decaf::util::StringTokenizer::toArray ( std::vector< std::string > & array ) [virtual]`

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

#### Parameters

<code>array</code>	- vector to place token strings in
--------------------	------------------------------------

#### Returns

number of string placed into the vector

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StringTokenizer.h`

## 6.786 `activemq::commands::SubscriptionInfo` Class Reference

```
#include <src/main/activemq/commands/SubscriptionInfo.h>
```

Inheritance diagram for `activemq::commands::SubscriptionInfo`:

### Public Member Functions

- **SubscriptionInfo** ()
- virtual **~SubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **SubscriptionInfo \* cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)

- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &**selector**)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &**subscriptionName**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &**subscribedDestination**)

### Static Public Attributes

- static const unsigned char **ID\_SUBSCRIPTIONINFO** = 55

### Protected Attributes

- std::string **clientId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::string **selector**
- std::string **subscriptionName**
- **Pointer**< **ActiveMQDestination** > **subscribedDestination**

## 6.786.1 Constructor & Destructor Documentation

6.786.1.1 **activemq::commands::SubscriptionInfo::SubscriptionInfo** ( )

6.786.1.2 **virtual activemq::commands::SubscriptionInfo::~~SubscriptionInfo** ( )  
[virtual]

## 6.786.2 Member Function Documentation

6.786.2.1 **virtual SubscriptionInfo\*** **activemq::commands::SubscriptionInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.786.2.2 `virtual void activemq::commands::SubscriptionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

6.786.2.3 `virtual bool activemq::commands::SubscriptionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

6.786.2.4 `virtual const std::string& activemq::commands::SubscriptionInfo::getClientId ( ) const [virtual]`

6.786.2.5 `virtual std::string& activemq::commands::SubscriptionInfo::getClientId ( ) [virtual]`

6.786.2.6 `virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.786.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ( ) const [virtual]`

6.786.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ( ) [virtual]`

- 6.786.2.9 `virtual const std::string& activemq::commands::SubscriptionInfo::getSelector ( )`  
`const [virtual]`
- 6.786.2.10 `virtual std::string& activemq::commands::SubscriptionInfo::getSelector ( )`  
`[virtual]`
- 6.786.2.11 `virtual std::string& activemq::commands::SubscriptionInfo::getSubscriptionName ( )`  
`[virtual]`
- 6.786.2.12 `virtual const std::string& activemq::commands::SubscriptionInfo::getSubscriptionName ( ) const` `[virtual]`
- 6.786.2.13 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ( ) const`  
`[virtual]`
- 6.786.2.14 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ( )`  
`[virtual]`
- 6.786.2.15 `virtual void activemq::commands::SubscriptionInfo::setClientId ( const std::string & clientId )` `[virtual]`
- 6.786.2.16 `virtual void activemq::commands::SubscriptionInfo::setDestination ( const Pointer< ActiveMQDestination > & destination )` `[virtual]`
- 6.786.2.17 `virtual void activemq::commands::SubscriptionInfo::setSelector ( const std::string & selector )` `[virtual]`
- 6.786.2.18 `virtual void activemq::commands::SubscriptionInfo::setSubscriptionName ( const std::string & subscriptionName )` `[virtual]`
- 6.786.2.19 `virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination ( const Pointer< ActiveMQDestination > & subscribedDestination )`  
`[virtual]`
- 6.786.2.20 `virtual std::string activemq::commands::SubscriptionInfo::toString ( ) const`  
`[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

### 6.786.3 Field Documentation

- 6.786.3.1 `std::string activemq::commands::SubscriptionInfo::clientId`  
[protected]
- 6.786.3.2 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination`  
[protected]
- 6.786.3.3 `const unsigned char activemq::commands::SubscriptionInfo::ID_SUBSCRIPTIONINFO = 55` [static]
- 6.786.3.4 `std::string activemq::commands::SubscriptionInfo::selector`  
[protected]
- 6.786.3.5 `std::string activemq::commands::SubscriptionInfo::subscriptionName`  
[protected]
- 6.786.3.6 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination`  
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SubscriptionInfo.h`

## 6.787 `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3620).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller`:

### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual `~SubscriptionInfoMarshaller` ()
- virtual `commands::DataStructure * createObject` () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char `getDataStructureType` () const  
*Get the Data Structure Type that identifies this Marshaler.*

- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.787.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3620).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.787.2 Constructor & Destructor Documentation

6.787.2.1 **activemq:wireformat:openwire:marshal:v3:SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.787.2.2 **virtual activemq:wireformat:openwire:marshal:v3:SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.787.3 Member Function Documentation

6.787.3.1 **virtual commands::DataStructure\* activemq:wireformat:openwire:marshal:v3:SubscriptionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq:wireformat:openwire:marshal::DataStreamMarshaller** (p. 1578).

```
6.787.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.787.3.3  virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.787.3.4  virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



## 6.787 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller

### Class Reference 3635

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.787.3.5 virtual int activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.787.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.787.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**SubscriptionInfoMarshaller.h**

## 6.788 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3624).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInf
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller:

#### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.788.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3624).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.788.2 Constructor & Destructor Documentation

6.788.2.1 **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.788.2.2 **virtual activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

## 6.788.3 Member Function Documentation

6.788.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::createObject** (  
 ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.788.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.788.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

6.788.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.788 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller

### Class Reference 3639

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.788.3.5 virtual int activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.788.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.788.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SubscriptionInfoMarshaller.h**

## 6.789 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3628).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInf
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**:

#### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.789.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3628).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.789.2 Constructor & Destructor Documentation

6.789.2.1 **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.789.2.2 **virtual activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

## 6.789.3 Member Function Documentation

6.789.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.789.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.789.3.3 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.789.3.4 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



## 6.789 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller

### Class Reference 3643

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.789.3.5 virtual int activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.789.3.6 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.789.3.7 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**SubscriptionInfoMarshaller.h**

## 6.790 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3632).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInf
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller:

#### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.790.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3632).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.790.2 Constructor & Destructor Documentation

6.790.2.1 **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
( ) [inline]

6.790.2.2 **virtual activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
( ) [inline, virtual]

## 6.790.3 Member Function Documentation

6.790.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.790.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.790.3.3  virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.790.3.4  virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.790 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller

### Class Reference 3647

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.790.3.5  virtual int activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.790.3.6  virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.790.3.7 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**SubscriptionInfoMarshaller.h**

## 6.791 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3636).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInf
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller**:

#### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.791.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3636).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.791.2 Constructor & Destructor Documentation

6.791.2.1 **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.791.2.2 **virtual activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

## 6.791.3 Member Function Documentation

6.791.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::createObject** (  
 ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.791.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.791.3.3  virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.791.3.4  virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--



## 6.791 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller

### Class Reference 3651

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.791.3.5 virtual int activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.791.3.6 virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.791.3.7 virtual void activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**SubscriptionInfoMarshaller.h**

## 6.792 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3640).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInf
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**:

#### Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

## 6.792.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3640).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.792.2 Constructor & Destructor Documentation

6.792.2.1 **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller**  
 ( ) [inline]

6.792.2.2 **virtual activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller**  
 ( ) [inline, virtual]

## 6.792.3 Member Function Documentation

6.792.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

## Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.792.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.792.3.3  virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.792.3.4  virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.792 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller

### Class Reference 3655

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.792.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

```
6.792.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

```
6.792.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**SubscriptionInfoMarshaller.h**

## 6.793 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

```
#include <src/main/decaf/util/concurrent/Synchronizable.h>
```

Inheritance diagram for decaf::util::concurrent::Synchronizable:

#### Public Member Functions

- virtual **~Synchronizable** ()
- virtual void **lock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to **Lock** (p. 2334) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0 throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **wait** (long long millisecs, int nanos)=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )

*Waits on a signal from this object, which is generated by a call to Notify.*

- virtual void **notify** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals a waiter on this object that it can now wake up and continue.*

- virtual void **notifyAll** ()=0 throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )

*Signals the waiters on this object that it can now wake up and continue.*

### 6.793.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Since

1.0

### 6.793.2 Constructor & Destructor Documentation

6.793.2.1 virtual decaf::util::concurrent::Synchronizable::~Synchronizable ( ) [inline, virtual]

### 6.793.3 Member Function Documentation

6.793.3.1 virtual void decaf::util::concurrent::Synchronizable::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Locks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implemented in **activemq::core::MessageDispatchChannel** (p. 2562), **decaf::internal::util::concurrent::Synchronizable** (p. 3656), **decaf::io::InputStream** (p. 2005), **decaf::io::OutputStream** (p. 2859), **decaf::util::AbstractCollection< E >** (p. 154), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1211), **decaf::util::concurrent::Mutex** (p. 2737), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3551), **decaf::util::StlQueue< T >** (p. 3560), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 154), **decaf::util::AbstractCollection< Pointer<**

**Synchronization** > > (p. 154), **decaf::util::AbstractCollection**< **Resource** \* > (p. 154),  
**decaf::util::AbstractCollection**< **cms::MessageConsumer** \* > (p. 154), **decaf::util::AbstractCollection**<  
**CompositeTask** \* > (p. 154), **decaf::util::AbstractCollection**< **URI** > (p. 154), **decaf::util::AbstractCollection**<  
**ActiveMQSession** \* > (p. 154), **decaf::util::AbstractCollection**< **Pointer**< **Desti-**  
**nationInfo** > > (p. 154), **decaf::util::AbstractCollection**< **PrimitiveValueNode** >  
(p. 154), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 154), **decaf::util::AbstractCollection**<  
**Pointer**< **BackupTransport** > > (p. 154), **decaf::util::AbstractCollection**< **cms::MessageProducer**  
\* > (p. 154), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 154), **decaf::util::AbstractCollection**<  
**cms::Session** \* > (p. 154), **decaf::util::AbstractCollection**< **cms::Connection** \*  
> > (p. 154), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >,  
**Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 1211), **decaf::util::concurrent::ConcurrentStlMap**<  
**Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR**  
> > (p. 1211), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId**  
>, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 1211), **decaf::util::concurrent::Concurr**  
**Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** >  
(p. 1211), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **LocalTransactionId**  
>, **Pointer**< **TransactionState** >, **LocalTransactionId::COMPARATOR** > (p. 1211),  
**decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **Pro-**  
**ducerState** >, **ProducerId::COMPARATOR** > (p. 1211), **decaf::util::StlMap**< **cms::Session**  
\*, **SessionResolver** \* > (p. 3551), **decaf::util::StlMap**< **std::string**, **WireFormat-**  
**Factory** \* > (p. 3551), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 3551),  
**decaf::util::StlMap**< **std::string**, **cms::Queue** \* > (p. 3551), **decaf::util::StlMap**<  
**Pointer**< **commands::ProducerId** >, **ActiveMQProducer** \*, **commands::ProducerId::COMPARATOR**  
> > (p. 3551), **decaf::util::StlMap**< **std::string**, **CachedConsumer** \* > (p. 3551),  
**decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** \*,  
**commands::ConsumerId::COMPARATOR** > (p. 3551), **decaf::util::StlMap**< **std::string**,  
**TransportFactory** \* > (p. 3551), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >,  
**Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 3551), **decaf::util::StlMap**<  
**int**, **Pointer**< **Command** > > (p. 3551), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId**  
>, **Dispatcher** \*, **commands::ConsumerId::COMPARATOR** > (p. 3551), **decaf::util::StlMap**<  
**std::string**, **CachedProducer** \* > (p. 3551), **decaf::util::StlMap**< **std::string**, **cms::Topic**  
\* > (p. 3551), **decaf::util::StlQueue**< **Pointer**< **Transport** > > (p. 3560), **decaf::util::StlQueue**<  
**Pointer**< **MessageDispatch** > > (p. 3560), **decaf::util::StlQueue**< **Task** > (p. 3560),  
**decaf::util::StlQueue**< **Pointer**< **Command** > > (p. 3560), and **decaf::util::StlQueue**<  
**decaf::lang::Pointer**< **commands::MessageDispatch** > > (p. 3560).

```

6.793.3.2  virtual void decaf::util::concurrent::Synchronizable::notify ( )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException ) [pure
            virtual]

```

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.



Implemented in `activemq::core::MessageDispatchChannel` (p. 2563), `decaf::internal::util::concurrent::Synchronizable` (p. 3656), `decaf::io::InputStream` (p. 2006), `decaf::io::OutputStream` (p. 2860), `decaf::util::AbstractCollection< E >` (p. 155), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1212), `decaf::util::concurrent::Mutex` (p. 2737), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 3551), `decaf::util::StlQueue< T >` (p. 3561), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 155), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 155), `decaf::util::AbstractCollection< Resource * >` (p. 155), `decaf::util::AbstractCollection< cms::MessageConsumer * >` (p. 155), `decaf::util::AbstractCollection< CompositeTask * >` (p. 155), `decaf::util::AbstractCollection< URI >` (p. 155), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 155), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 155), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 155), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 155), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 155), `decaf::util::AbstractCollection< cms::MessageProducer * >` (p. 155), `decaf::util::AbstractCollection< cms::Destination * >` (p. 155), `decaf::util::AbstractCollection< cms::Session * >` (p. 155), `decaf::util::AbstractCollection< cms::Connection * >` (p. 155), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, MessageId::COMPARATOR >` (p. 1212), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1212), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1212), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1212), `decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >` (p. 1212), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1212), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 3551), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 3551), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 3551), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 3551), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 3551), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 3551), `decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< int, Pointer< Command > >` (p. 3551), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 3551), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 3551), `decaf::util::StlMap< std::string, cms::Topic * >` (p. 3551), `decaf::util::StlQueue< Pointer< Transport > >` (p. 3561), `decaf::util::StlQueue< Pointer< MessageDispatch > >` (p. 3561), `decaf::util::StlQueue< Task >` (p. 3561), `decaf::util::StlQueue< Pointer< Command > >` (p. 3561), and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >` (p. 3561).

```
6.793.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll ( )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException ) [pure
            virtual]
```

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2563), **decaf::internal::util::concurrent::Synchronizable** (p. 3656), **decaf::io::InputStream** (p. 2007), **decaf::io::OutputStream** (p. 2860), **decaf::util::AbstractCollection** (p. 155), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1212), **decaf::util::concurrent::Mutex** (p. 2738), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3552), **decaf::util::StlQueue< T >** (p. 3561), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 155), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 155), **decaf::util::AbstractCollection< Resource \* >** (p. 155), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 155), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 155), **decaf::util::AbstractCollection< URI >** (p. 155), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 155), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 155), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 155), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 155), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 155), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 155), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 155), **decaf::util::AbstractCollection< cms::Session \* >** (p. 155), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 155), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1212), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1212), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1212), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1212), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1212), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1212), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3552), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3552), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3552), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3552), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3552), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3552), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3552), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3552), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3552), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3552), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3552), **decaf::util::StlMap< std::string, CachedProducer \* >** (p. 3552), **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3552), **decaf::util::StlQueue< Pointer< Transport > >** (p. 3561), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 3561), **decaf::util::StlQueue< Task >** (p. 3561), **decaf::util::StlQueue< Pointer< Command > >** (p. 3561), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 3561).

6.793.3.4 virtual bool decaf::util::concurrent::Synchronizable::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [pure virtual]

Attempts to **Lock** (p. 2334) the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implemented in **activemq::core::MessageDispatchChannel** (p. 2564), **decaf::internal::util::concurrent::Synchronizable** (p. 3657), **decaf::io::InputStream** (p. 2011), **decaf::io::OutputStream** (p. 2861), **decaf::util::AbstractCollection< E >** (p. 158), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1217), **decaf::util::concurrent::Mutex** (p. 2738), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3554), **decaf::util::StlQueue< T >** (p. 3562), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 158), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 158), **decaf::util::AbstractCollection< Resource \* >** (p. 158), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 158), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 158), **decaf::util::AbstractCollection< URI >** (p. 158), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 158), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 158), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 158), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 158), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 158), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 158), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 158), **decaf::util::AbstractCollection< cms::Session \* >** (p. 158), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 158), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1217), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3554), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3554), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3554), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3554), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3554), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3554), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3554), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3554), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3554), **decaf::util::StlMap< int, Pointer< Command > >** (p. 3554), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR >** (p. 3554), **decaf::util::StlMap<**

**std::string, CachedProducer \* >** (p. 3554), **decaf::util::StlMap< std::string, cms::Topic \* >** (p. 3554), **decaf::util::StlQueue< Pointer< Transport > >** (p. 3562), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 3562), **decaf::util::StlQueue< Task >** (p. 3562), **decaf::util::StlQueue< Pointer< Command > >** (p. 3562), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 3562).

6.793.3.5 **virtual void decaf::util::concurrent::Synchronizable::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException )** [pure virtual]

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implemented in **activemq::core::MessageDispatchChannel** (p. 2564), **decaf::internal::util::concurrent::Synchronizable** (p. 3657), **decaf::io::InputStream** (p. 2011), **decaf::io::OutputStream** (p. 2861), **decaf::util::AbstractCollection< E >** (p. 159), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1217), **decaf::util::concurrent::Mutex** (p. 2739), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3554), **decaf::util::StlQueue< T >** (p. 3563), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 159), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 159), **decaf::util::AbstractCollection< Resource \* >** (p. 159), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 159), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 159), **decaf::util::AbstractCollection< URI >** (p. 159), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 159), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 159), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 159), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 159), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 159), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 159), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 159), **decaf::util::AbstractCollection< cms::Session \* >** (p. 159), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 159), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1217), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1217), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3554), **decaf::util::StlMap< std::string, WireFormatFactory \* >** (p. 3554), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 3554), **decaf::util::StlMap< std::string, cms::Queue \* >** (p. 3554), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR >** (p. 3554), **decaf::util::StlMap< std::string, CachedConsumer \* >** (p. 3554), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR >** (p. 3554), **decaf::util::StlMap< std::string, TransportFactory \* >** (p. 3554), **decaf::util::StlMap< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR >** (p. 3554), **decaf::util::StlMap<**

int, Pointer< Command > > (p. 3554), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3554), decaf::util::StlMap< std::string, CachedProducer \* > (p. 3554), decaf::util::StlMap< std::string, cms::Topic \* > (p. 3554), decaf::util::StlQueue< Pointer< Transport > > (p. 3563), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 3563), decaf::util::StlQueue< Task > (p. 3563), decaf::util::StlQueue< Pointer< Command > > (p. 3563), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3563).

```
6.793.3.6 virtual void decaf::util::concurrent::Synchronizable::wait ( )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException,
                    decaf::lang::exceptions::InterruptedException ) [pure virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2566), **decaf::internal::util::concurrent::Synchronizable** (p. 3657), **decaf::io::InputStream** (p. 2012), **decaf::io::OutputStream** (p. 2861), **decaf::util::AbstractCollection< E >** (p. 159), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1218), **decaf::util::concurrent::Mutex** (p. 2739), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 3555), **decaf::util::StlQueue< T >** (p. 3563), **decaf::util::AbstractCollection< transport::TransportListener \* >** (p. 159), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 159), **decaf::util::AbstractCollection< Resource \* >** (p. 159), **decaf::util::AbstractCollection< cms::MessageConsumer \* >** (p. 159), **decaf::util::AbstractCollection< CompositeTask \* >** (p. 159), **decaf::util::AbstractCollection< URI >** (p. 159), **decaf::util::AbstractCollection< ActiveMQSession \* >** (p. 159), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 159), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 159), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 159), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 159), **decaf::util::AbstractCollection< cms::MessageProducer \* >** (p. 159), **decaf::util::AbstractCollection< cms::Destination \* >** (p. 159), **decaf::util::AbstractCollection< cms::Session \* >** (p. 159), **decaf::util::AbstractCollection< cms::Connection \* >** (p. 159), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1218), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1218), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1218), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1218), **decaf::util::concurrent::ConcurrentStlMap< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR >** (p. 1218), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1218), **decaf::util::StlMap< cms::Session \*, SessionResolver \* >** (p. 3555), **decaf::util::StlMap< std::string, WireFormat-**

**Factory** \* > (p. 3555), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 3555), **decaf::util::StlMap**< **std::string**, **cms::Queue** \* > (p. 3555), **decaf::util::StlMap**< **Pointer**< **commands::ProducerId** >, **ActiveMQProducer** \*, **commands::ProducerId::COMPARATOR** > (p. 3555), **decaf::util::StlMap**< **std::string**, **CachedConsumer** \* > (p. 3555), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** \*, **commands::ConsumerId::COMPARATOR** > (p. 3555), **decaf::util::StlMap**< **std::string**, **TransportFactory** \* > (p. 3555), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 3555), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 3555), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **Dispatcher** \*, **commands::ConsumerId::COMPARATOR** > (p. 3555), **decaf::util::StlMap**< **std::string**, **CachedProducer** \* > (p. 3555), **decaf::util::StlMap**< **std::string**, **cms::Topic** \* > (p. 3555), **decaf::util::StlQueue**< **Pointer**< **Transport** > > (p. 3563), **decaf::util::StlQueue**< **Pointer**< **MessageDispatch** > > (p. 3563), **decaf::util::StlQueue**< **Task** > (p. 3563), **decaf::util::StlQueue**< **Pointer**< **Command** > > (p. 3563), and **decaf::util::StlQueue**< **decaf::lang::Pointer**< **commands::MessageDispatch** > > (p. 3563).

```
6.793.3.7 virtual void decaf::util::concurrent::Synchronizable::wait ( long long
    millisecs ) throw ( decaf::lang::exceptions::RuntimeException,
    decaf::lang::exceptions::IllegalMonitorStateException,
    decaf::lang::exceptions::InterruptedException ) [pure virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2565), **decaf::internal::util::concurrent::Synchronizable** (p. 3658), **decaf::io::InputStream** (p. 2011), **decaf::io::OutputStream** (p. 2862), **decaf::util::AbstractCollection** (p. 160), **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 1218), **decaf::util::concurrent::Mutex** (p. 2740), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 3556), **decaf::util::StlQueue**< **T** > (p. 3563), **decaf::util::AbstractCollection**< **transport::TransportListener** \* > (p. 160), **decaf::util::AbstractCollection**< **Pointer**< **Synchronization** > > (p. 160), **decaf::util::AbstractCollection**< **Resource** \* > (p. 160), **decaf::util::AbstractCollection**< **cms::MessageConsumer** \* > (p. 160), **decaf::util::AbstractCollection**< **CompositeTask** \* > (p. 160), **decaf::util::AbstractCollection**< **URI** > (p. 160), **decaf::util::AbstractCollection**< **ActiveMQSession** \* > (p. 160), **decaf::util::AbstractCollection**< **Pointer**< **DestinationInfo** > > (p. 160), **decaf::util::AbstractCollection**< **PrimitiveValueNode** > (p. 160), **decaf::util::AbstractCollection**< **Pointer**< **Command** > > (p. 160), **decaf::util::AbstractCollection**< **Pointer**< **BackupTransport** > > (p. 160), **decaf::util::AbstractCollection**< **cms::MessageProducer**

\* > (p. 160), **decaf::util::AbstractCollection**< **cms::Destination** \* > (p. 160), **decaf::util::AbstractCollection**< **cms::Session** \* > (p. 160), **decaf::util::AbstractCollection**< **cms::Connection** \* > (p. 160), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >, **Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 1218), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR** > (p. 1218), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 1218), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 1218), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **LocalTransactionId** >, **Pointer**< **TransactionState** >, **LocalTransactionId::COMPARATOR** > (p. 1218), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** > (p. 1218), **decaf::util::StlMap**< **cms::Session** \*, **SessionResolver** \* > (p. 3556), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** \* > (p. 3556), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 3556), **decaf::util::StlMap**< **std::string**, **cms::Queue** \* > (p. 3556), **decaf::util::StlMap**< **Pointer**< **commands::ProducerId** >, **ActiveMQProducer** \*, **commands::ProducerId::COMPARATOR** > (p. 3556), **decaf::util::StlMap**< **std::string**, **CachedConsumer** \* > (p. 3556), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** \*, **commands::ConsumerId::COMPARATOR** > (p. 3556), **decaf::util::StlMap**< **std::string**, **TransportFactory** \* > (p. 3556), **decaf::util::StlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerInfo** >, **ConsumerId::COMPARATOR** > (p. 3556), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 3556), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **Dispatcher** \*, **commands::ConsumerId::COMPARATOR** > (p. 3556), **decaf::util::StlMap**< **std::string**, **CachedProducer** \* > (p. 3556), **decaf::util::StlMap**< **std::string**, **cms::Topic** \* > (p. 3556), **decaf::util::StlQueue**< **Pointer**< **Transport** > > (p. 3563), **decaf::util::StlQueue**< **Pointer**< **MessageDispatch** > > (p. 3563), **decaf::util::StlQueue**< **Task** > (p. 3563), **decaf::util::StlQueue**< **Pointer**< **Command** > > (p. 3563), and **decaf::util::StlQueue**< **decaf::lang::Pointer**< **commands::MessageDispatch** > > (p. 3563).

**6.793.3.8** `virtual void decaf::util::concurrent::Synchronizable::wait ( long long milliseconds, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [pure virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>milliseconds</i>	the time in milliseconds to wait, or <code>WAIT_INFINITE</code>
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

## Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the <b>Synchronizable</b> (p. 3644) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2565), **decaf::internal::util::concurrent::Synchronizable** (p. 3658), **decaf::io::InputStream** (p. 2012), **decaf::io::OutputStream** (p. 2862), **decaf::util::AbstractCollection** (p. 159), **decaf::util::concurrent::ConcurrentStlMap**< K, V, COMPARATOR > (p. 1219), **decaf::util::concurrent::Mutex** (p. 2740), **decaf::util::StlMap**< K, V, COMPARATOR > (p. 3555), **decaf::util::StlQueue**< T > (p. 3564), **decaf::util::AbstractCollection**< transport::TransportListener \* > (p. 159), **decaf::util::AbstractCollection**< Pointer< Synchronization > > (p. 159), **decaf::util::AbstractCollection**< Resource \* > (p. 159), **decaf::util::AbstractCollection**< cms::MessageConsumer \* > (p. 159), **decaf::util::AbstractCollection**< CompositeTask \* > (p. 159), **decaf::util::AbstractCollection**< URI > (p. 159), **decaf::util::AbstractCollection**< ActiveMQSession \* > (p. 159), **decaf::util::AbstractCollection**< Pointer< DestinationInfo > > (p. 159), **decaf::util::AbstractCollection**< PrimitiveValueNode > (p. 159), **decaf::util::AbstractCollection**< Pointer< Command > > (p. 159), **decaf::util::AbstractCollection**< Pointer< BackupTransport > > (p. 159), **decaf::util::AbstractCollection**< cms::MessageProducer \* > (p. 159), **decaf::util::AbstractCollection**< cms::Destination \* > (p. 159), **decaf::util::AbstractCollection**< cms::Session \* > (p. 159), **decaf::util::AbstractCollection**< cms::Connection \* > (p. 159), **decaf::util::concurrent::ConcurrentStlMap**< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1219), **decaf::util::concurrent::ConcurrentStlMap**< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1219), **decaf::util::concurrent::ConcurrentStlMap**< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1219), **decaf::util::concurrent::ConcurrentStlMap**< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1219), **decaf::util::concurrent::ConcurrentStlMap**< Pointer< LocalTransactionId >, Pointer< TransactionState >, LocalTransactionId::COMPARATOR > (p. 1219), **decaf::util::concurrent::ConcurrentStlMap**< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1219), **decaf::util::StlMap**< cms::Session \*, SessionResolver \* > (p. 3555), **decaf::util::StlMap**< std::string, WireFormatFactory \* > (p. 3555), **decaf::util::StlMap**< std::string, PrimitiveValueNode > (p. 3555), **decaf::util::StlMap**< std::string, cms::Queue \* > (p. 3555), **decaf::util::StlMap**< Pointer< commands::ProducerId >, ActiveMQProducer \*, commands::ProducerId::COMPARATOR > (p. 3555), **decaf::util::StlMap**< std::string, CachedConsumer \* > (p. 3555), **decaf::util::StlMap**< Pointer< commands::ConsumerId >, ActiveMQConsumer \*, commands::ConsumerId::COMPARATOR > (p. 3555), **decaf::util::StlMap**< std::string, TransportFactory \* > (p. 3555), **decaf::util::StlMap**< Pointer< ConsumerId >, Pointer< ConsumerInfo >, ConsumerId::COMPARATOR > (p. 3555), **decaf::util::StlMap**< int, Pointer< Command > > (p. 3555), **decaf::util::StlMap**< Pointer< commands::ConsumerId >, Dispatcher \*, commands::ConsumerId::COMPARATOR > (p. 3555), **decaf::util::StlMap**< std::string, CachedProducer \* > (p. 3555), **decaf::util::StlMap**< std::string, cms::Topic \* > (p. 3555), **decaf::util::StlQueue**< Pointer< Transport > > (p. 3564), **decaf::util::StlQueue**< Pointer< MessageDispatch > > (p. 3564), **decaf::util::StlQueue**< Task > (p. 3564), **decaf::util::StlQueue**< Pointer< Command > > (p. 3564), and **decaf::util::StlQueue**< decaf::lang::Pointer< commands::MessageDispatch > > (p. 3564).



## 6.794 decaf::internal::util::concurrent::SynchronizableImpl Class Reference 3667

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Synchronizable.h**

## 6.794 decaf::internal::util::concurrent::SynchronizableImpl Class Reference

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

```
#include <src/main/decaf/internal/util/concurrent/SynchronizableImpl.h>
```

Inheritance diagram for decaf::internal::util::concurrent::SynchronizableImpl:

### Public Member Functions

- **SynchronizableImpl** ()
- virtual ~**SynchronizableImpl** ()
- virtual void **lock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Locks the object.*
- virtual bool **tryLock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Attempts to Lock the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw ( decaf::lang::exceptions::RuntimeException )  
*Unlocks the object.*
- virtual void **wait** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **wait** (long long millisecs, int nanos) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException )  
*Waits on a signal from this object, which is generated by a call to Notify.*
- virtual void **notify** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals a waiter on this object that it can now wake up and continue.*
- virtual void **notifyAll** () throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException )  
*Signals the waiters on this object that it can now wake up and continue.*

### 6.794.1 Detailed Description

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

#### Since

1.0

### 6.794.2 Constructor & Destructor Documentation

6.794.2.1 `decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl ( )`

6.794.2.2 `virtual decaf::internal::util::concurrent::SynchronizableImpl::~~SynchronizableImpl ( ) [virtual]`

### 6.794.3 Member Function Documentation

6.794.3.1 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]`

Locks the object.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3645).

6.794.3.2 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

#### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3646).

## 6.794 decaf::internal::util::concurrent::SynchronizableImpl Class Reference 3669

6.794.3.3 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notifyAll ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException ) [virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

### Exceptions

<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.
<i>RuntimeException</i>	if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 3647).

6.794.3.4 virtual bool decaf::internal::util::concurrent::SynchronizableImpl::tryLock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

### Returns

true if the lock was acquired, false if it is already held by another thread.

### Exceptions

<i>RuntimeException</i>	if an error occurs while locking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3649).

6.794.3.5 virtual void decaf::internal::util::concurrent::SynchronizableImpl::unlock ( ) throw ( decaf::lang::exceptions::RuntimeException ) [virtual]

Unlocks the object.

### Exceptions

<i>RuntimeException</i>	if an error occurs while unlocking the object.
-------------------------	--

Implements **decaf::util::concurrent::Synchronizable** (p. 3650).

6.794.3.6 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait ( ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3651).

6.794.3.7 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait ( long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
------------------	--

#### Exceptions

<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3652).

6.794.3.8 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait ( long long millisecs, int nanos ) throw ( decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException ) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified

time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

#### Parameters

<i>millisecs</i>	the time in milliseconds to wait, or WAIT_INFINITE
<i>nanos</i>	additional time in nanoseconds with a range of 0-999999

#### Exceptions

<i>IllegalArgumentException</i>	if an error occurs or the nanos argument is not in the range of [0-999999]
<i>RuntimeException</i>	if an error occurs while waiting on the object.
<i>InterruptedException</i>	if the wait is interrupted before it completes.
<i>IllegalMonitorStateException</i>	- if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 3653).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**SynchronizableImpl.h**

## 6.795 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 3659), used to sync the events of a Transaction with the items in the Transaction.

```
#include <src/main/activemq/core/Synchronization.h>
```

#### Public Member Functions

- virtual **~Synchronization** ()
- virtual void **beforeEnd** ()=0 throw ( exceptions::ActiveMQException )
- virtual void **afterCommit** ()=0 throw ( exceptions::ActiveMQException )
- virtual void **afterRollback** ()=0 throw ( exceptions::ActiveMQException )

#### 6.795.1 Detailed Description

Transacted Object **Synchronization** (p. 3659), used to sync the events of a Transaction with the items in the Transaction.

## 6.795.2 Constructor & Destructor Documentation

6.795.2.1 `virtual activemq::core::Synchronization::~~Synchronization ( ) [inline, virtual]`

## 6.795.3 Member Function Documentation

6.795.3.1 `virtual void activemq::core::Synchronization::afterCommit ( ) throw ( exceptions::ActiveMQException ) [pure virtual]`

6.795.3.2 `virtual void activemq::core::Synchronization::afterRollback ( ) throw ( exceptions::ActiveMQException ) [pure virtual]`

6.795.3.3 `virtual void activemq::core::Synchronization::beforeEnd ( ) throw ( exceptions::ActiveMQException ) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Synchronization.h`

## 6.796 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

A **blocking queue** (p. 804) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

```
#include <src/main/decaf/util/concurrent/SynchronousQueue.h>
```

Inheritance diagram for `decaf::util::concurrent::SynchronousQueue< E >`:

### Data Structures

- class `EmptyIterator`

### Public Member Functions

- `SynchronousQueue ( )`
- `virtual ~SynchronousQueue ( )`
- `virtual void put (const E &value) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )`

*Adds the specified element to this queue, waiting if necessary for another thread to receive it.*

- virtual bool **offer** (const E &e, long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.*
- virtual bool **offer** (const E &value) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException )  
*Inserts the specified element into this queue, if another thread is waiting to receive it.*
- virtual E **take** () throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.*
- virtual bool **poll** (E &result, long long timeout, const **TimeUnit** &unit) throw ( decaf::lang::exceptions::InterruptedException )  
*Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.*
- virtual bool **poll** (E &result)  
*Retrieves and removes the head of this queue, if another thread is currently making an element available.*
- virtual bool **equals** (const **Collection**< E > &value) const  
*Answers true if this **Collection** (p. 1155) and the one given are the same size and if each element contained in the **Collection** (p. 1155) given is equal to an element contained in this collection.*
- virtual **decaf::util::Iterator**< E > \* **iterator** ()
- virtual **decaf::util::Iterator**< E > \* **iterator** () const
- virtual bool **isEmpty** () const  
*Returns true if this collection contains no elements.*
- virtual std::size\_t **size** () const  
*Returns the number of elements in this collection.*
- virtual int **remainingCapacity** () const  
*Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or Integer::MAX\_VALUE if there is no intrinsic limit.*
- virtual void **clear** () throw ( lang::exceptions::UnsupportedOperationException )  
*Removes all elements of the queue.*
- virtual bool **contains** (const E &value DECAF\_UNUSED) const throw ( lang::Exception )
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw ( lang::Exception )  
*Returns true if this collection contains all of the elements in the specified collection.*
- virtual bool **remove** (const E &value DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **removeAll** (const **Collection**< E > &collection DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )
- virtual bool **retainAll** (const **Collection**< E > &collection DECAF\_UNUSED) throw ( lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException )

- virtual bool **peek** (E &result DECAF\_UNUSED) const
- virtual std::vector< E > **toArray** () const

*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1155).*

- virtual std::size\_t **drainTo** (**Collection**< E > &c) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )

*Removes all available elements from this queue and adds them to the given collection.*

- virtual std::size\_t **drainTo** (**Collection**< E > &c, std::size\_t maxElements) throw ( decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException )

*Removes at most the given number of available elements from this queue and adds them to the given collection.*

### 6.796.1 Detailed Description

template<typename E>class decaf::util::concurrent::SynchronousQueue< E >

A **blocking queue** (p. 804) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

A synchronous queue does not have any internal capacity, not even a capacity of one. You cannot **peek** at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and **poll()** (p. 3667) will return `null`. For purposes of other **Collection** (p. 1155) methods (for example **contains**), a **SynchronousQueue** (p. 3660) acts as an empty collection. This queue does not permit `null` elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to `true` grants threads access in FIFO order.

This class and its iterator implement all of the *optional* methods of the **Collection** (p. 1155) and **Iterator** (p. 2114) interfaces.

#### Since

1.0

### 6.796.2 Constructor & Destructor Documentation



6.796.2.1 `template<typename E > decaf::util::concurrent::SynchronousQueue< E  
>::SynchronousQueue( ) [inline]`

6.796.2.2 `template<typename E > virtual decaf::util::concurrent::SynchronousQueue<  
E>::~SynchronousQueue( ) [inline, virtual]`

### 6.796.3 Member Function Documentation

6.796.3.1 `template<typename E > virtual void  
decaf::util::concurrent::SynchronousQueue< E  
>::clear( ) throw ( lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractQueue< E >** (p. 166).

6.796.3.2 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::contains( const E &value DECAF_UNUSED ) const throw ( lang::Exception )  
[inline, virtual]`

6.796.3.3 `template<typename E > virtual bool  
decaf::util::concurrent::SynchronousQueue< E  
>::containsAll( const Collection< E > &collection ) const throw (   
lang::Exception ) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

#### Parameters

<i>collection</i>	collection to be checked for containment in this collection
-------------------	---

#### Returns

true if this collection contains all of the elements in the specified collection.

#### Exceptions

<i>Exception</i>	if an error occurs,
------------------	---------------------

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 153).

```

6.796.3.4  template<typename E > virtual std::size_t
           decaf::util::concurrent::SynchronousQueue<
           E >::drainTo ( Collection< E > & c ) throw ( de-
           caf::lang::exceptions::UnsupportedOperationException,
           decaf::lang::exceptions::IllegalArgumentException ) [inline,
           virtual]

```

Removes all available elements from this queue and adds them to the given collection.

This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

#### Parameters

<i>c</i>	the collection to transfer elements into
----------	--

#### Returns

the number of elements transferred

#### Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 807).

References `decaf::util::AbstractQueue< E >::element()`, and `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

```

6.796.3.5  template<typename E > virtual std::size_t
           decaf::util::concurrent::SynchronousQueue< E
           >::drainTo ( Collection< E > & c, std::size_t maxElements ) throw
           ( decaf::lang::exceptions::UnsupportedOperationException,
           decaf::lang::exceptions::IllegalArgumentException ) [inline,
           virtual]

```

Removes at most the given number of available elements from this queue and adds them to the given collection.

A failure encountered while attempting to add elements to collection *c* may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in `IllegalArgumentException`. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

#### Parameters

<i>c</i>	the collection to transfer elements into
<i>maxElements</i>	the maximum number of elements to transfer

#### Returns

the number of elements transferred

#### Exceptions

<i>UnsupportedOperationException</i>	if addition of elements is not supported by the specified collection
<i>IllegalArgumentException</i>	if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

Implements **decaf::util::concurrent::BlockingQueue< E >** (p. 807).

References **decaf::util::AbstractQueue< E >::element()**, and **decaf::util::concurrent::SynchronousQueue< E >::poll()**.

```
6.796.3.6  template<typename E > virtual bool
           decaf::util::concurrent::SynchronousQueue< E
           >::equals ( const Collection< E > & collection ) const  [inline,
           virtual]
```

Answers true if this **Collection** (p. 1155) and the one given are the same size and if each element contained in the **Collection** (p. 1155) given is equal to an element contained in this collection.

#### Parameters

<i>collection</i>	- The <b>Collection</b> (p. 1155) to be compared to this one.
-------------------	---

#### Returns

true if this **Collection** (p. 1155) is equal to the one given.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 153).

```
6.796.3.7  template<typename E > virtual bool
           decaf::util::concurrent::SynchronousQueue< E
           >::isEmpty ( ) const  [inline, virtual]
```

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 3669) == 0.

#### Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection**< E > (p. 154).

```
6.796.3.8  template<typename E > virtual decaf::util::Iterator<E>*
           decaf::util::concurrent::SynchronousQueue< E >::iterator ( ) const
           [inline, virtual]
```

Implements **decaf::lang::Iterable**< E > (p. 2114).

```
6.796.3.9  template<typename E > virtual decaf::util::Iterator<E>*
           decaf::util::concurrent::SynchronousQueue< E >::iterator ( )
           [inline, virtual]
```

### Returns

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable**< E > (p. 2113).

```
6.796.3.10 template<typename E > virtual bool
           decaf::util::concurrent::SynchronousQueue< E
           >::offer ( const E & e, long timeout, const TimeUnit & unit )
           throw ( decaf::lang::exceptions::InterruptedException,
           decaf::lang::exceptions::NullPointerException,
           decaf::lang::exceptions::IllegalArgumentException ) [inline,
           virtual]
```

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

### Returns

`true` if successful, or `false` if the specified waiting time elapses before a consumer appears.

### Exceptions

<i>InterruptedException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
<i>NullPointerException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.
<i>IllegalArgumentException</i>	Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue**< E > (p. 808).

## 6.796 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

3679

```
6.796.3.11  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E >::offer ( const E
             & value ) throw ( decaf::lang::exceptions::NullPointerException,
             decaf::lang::exceptions::IllegalArgumentException ) [inline,
             virtual]
```

Inserts the specified element into this queue, if another thread is waiting to receive it.

### Parameters

<i>value</i>	the element to add to the <b>Queue</b> (p. 3094)
--------------	--

### Returns

true if the element was added to this queue, else false

### Exceptions

<i>NullPointerException</i>	if the <b>Queue</b> (p. 3094) implementation does not allow Null values to be inserted into the <b>Queue</b> (p. 3094).
<i>IllegalArgumentException</i>	if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 3096).

```
6.796.3.12  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::peek ( E &result DECAF_UNUSED ) const [inline, virtual]
```

```
6.796.3.13  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::poll ( E &result, long long timeout, const TimeUnit & unit ) throw
             ( decaf::lang::exceptions::InterruptedException ) [inline,
             virtual]
```

Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.

### Parameters

<i>result</i>	a reference to the value where the head of the <b>Queue</b> (p. 3094) should be copied to.
<i>timeout</i>	the time that the method should block if there is no element available to return.
<i>unit</i>	the Time Units that the timeout value represents.

### Returns

true if the head of the **Queue** (p. 3094) was copied to the result param or false if no value could be returned.

Implements **decaf::util::concurrent::BlockingQueue**< **E** > (p. 809).

Referenced by decaf::util::concurrent::SynchronousQueue< **E** >::drainTo().

```
6.796.3.14  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::poll ( E & result ) [inline, virtual]
```

Retrieves and removes the head of this queue, if another thread is currently making an element available.

#### Parameters

<i>result</i>	a reference to the value where the head of the <b>Queue</b> (p. 3094) should be copied to.
---------------	--

#### Returns

true if the head of the **Queue** (p. 3094) was copied to the result param or false if no value could be returned.

Implements **decaf::util::Queue**< **E** > (p. 3097).

```
6.796.3.15  template<typename E > virtual void
             decaf::util::concurrent::SynchronousQueue< E >::put
             ( const E & value ) throw ( decaf::lang::exceptions::InterruptedException,
             decaf::lang::exceptions::NullPointerException,
             decaf::lang::exceptions::IllegalArgumentException ) [inline,
             virtual]
```

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

#### Parameters

<i>value</i>	the element to add to the <b>Queue</b> (p. 3094).
--------------	---

#### Exceptions

<i>InterruptedException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.
<i>NullPointerException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.
<i>IllegalArgumentException</i>	Inserts the specified element into this queue, waiting if necessary for space to become available.

Implements **decaf::util::concurrent::BlockingQueue**< **E** > (p. 809).

```
6.796.3.16  template<typename E > virtual int
             decaf::util::concurrent::SynchronousQueue< E
             >::remainingCapacity ( ) const  [inline, virtual]
```

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking, or `Integer::MAX_VALUE` if there is no intrinsic limit.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting `remainingCapacity` because it may be the case that another thread is about to insert or remove an element.

### Returns

the remaining capacity

Implements `decaf::util::concurrent::BlockingQueue< E >` (p. 810).

```
6.796.3.17  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::remove ( const E &value DECAF_UNUSED ) throw (
             lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline,
             virtual]
```

```
6.796.3.18  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::removeAll ( const Collection< E > &collection DECAF_UNUSED )
             throw ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline,
             virtual]
```

```
6.796.3.19  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::retainAll ( const Collection< E > &collection DECAF_UNUSED )
             throw ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline,
             virtual]
```

```
6.796.3.20  template<typename E > virtual std::size_t
             decaf::util::concurrent::SynchronousQueue< E
             >::size ( ) const  [inline, virtual]
```

Returns the number of elements in this collection.

If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

### Returns

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 1164).

```
6.796.3.21  template<typename E > virtual E
             decaf::util::concurrent::SynchronousQueue< E
             >::take ( ) throw ( decaf::lang::exceptions::InterruptedException )
             [inline, virtual]
```

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

### Returns

the head of this queue

### Exceptions

<i>InterruptedException</i>	Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.
-----------------------------	--

Implements **decaf::util::concurrent::BlockingQueue**< **E** > (p. 810).

```
6.796.3.22  template<typename E > virtual std::vector<E>
             decaf::util::concurrent::SynchronousQueue< E >::toArray ( ) const
             [inline, virtual]
```

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 1155).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

### Returns

an vector of copies of all the elements from this **Collection** (p. 1155)

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 158).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**SynchronousQueue.h**

## 6.797 decaf::lang::System Class Reference

The **System** (p. 3670) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

```
#include <src/main/decaf/lang/System.h>
```



## Public Member Functions

- virtual `~System ()`

## Static Public Member Functions

- static void **arraycopy** (const unsigned char \*src, std::size\_t srcPos, unsigned char \*dest, std::size\_t destPos, std::size\_t length)

*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*

- static void **arraycopy** (const short \*src, std::size\_t srcPos, short \*dest, std::size\_t destPos, std::size\_t length)

*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*

- static void **arraycopy** (const int \*src, std::size\_t srcPos, int \*dest, std::size\_t destPos, std::size\_t length)

*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*

- static void **arraycopy** (const long long \*src, std::size\_t srcPos, long long \*dest, std::size\_t destPos, std::size\_t length)

*Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.*

- static const **util::Map**< std::string, std::string > & **getenv ()**

*Enumerates the system environment and returns a map of env variable names to the string values they hold.*

- static std::string **getenv** (const std::string &name)

*Reads an environment value from the system and returns it as a string object.*

- static void **unsetenv** (const std::string &name)

*Clears a set environment value if one is set.*

- static void **setenv** (const std::string &name, const std::string &value)

*Sets the specified system property to the value given.*

- static long long **currentTimeMillis** ()

*Returns the current time in milliseconds.*

- static long long **nanoTime** ()

*Returns the current value of the most precise available system timer, in nanoseconds.*

- static int **availableProcessors** ()

*Returns the number of processors available for execution of Decaf Threads.*

- static **decaf::util::Properties** & **getProperties** ()

*Gets the Properties object that holds the Properties accessed from calls to getProperty and setProperty.*

- static std::string **getProperty** (const std::string &key)

*Gets the specified **System** (p. 3670) property if set, otherwise returns an empty string.*

- static std::string **getProperty** (const std::string &key, const std::string &default-Value)  
*Gets the specified **System** (p. 3670) property if set, otherwise returns the specified default value.*
- static std::string **setProperty** (const std::string &key, const std::string &value)  
*Sets the **System** (p. 3670) Property to the specified value.*
- static std::string **clearProperty** (const std::string &key)  
*Clear any value associated with the system property specified.*

### Protected Member Functions

- **System** ()

### Friends

- class **decaf::lang::Runtime**

## 6.797.1 Detailed Description

The **System** (p. 3670) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.

### Since

1.0

## 6.797.2 Constructor & Destructor Documentation

6.797.2.1 **decaf::lang::System::System** ( ) [protected]

6.797.2.2 **virtual decaf::lang::System::~~System** ( ) [inline, virtual]

## 6.797.3 Member Function Documentation

6.797.3.1 **static void decaf::lang::System::arraycopy** ( const unsigned char \* *src*, std::size\_t *srcPos*, unsigned char \* *dest*, std::size\_t *destPos*, std::size\_t *length* ) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

### Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.

<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

**Exceptions**

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

Referenced by decaf::lang::ArrayPointer< unsigned char >::clone().

**6.797.3.2** static void decaf::lang::System::arraycopy ( const short \* *src*, std::size\_t *srcPos*, short \* *dest*, std::size\_t *destPos*, std::size\_t *length* ) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

**Parameters**

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

**Exceptions**

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

**6.797.3.3** static void decaf::lang::System::arraycopy ( const long long \* *src*, std::size\_t *srcPos*, long long \* *dest*, std::size\_t *destPos*, std::size\_t *length* ) [static]

Copies the number of elements specified by length from the source array starting at the given source offset specified by srcPos to the dest array starting at the given destination offset given by destPos.

**Parameters**

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from src to dest.

**Exceptions**

<i>NullPointerException</i>	if src or dest are NULL.
-----------------------------	--------------------------

**6.797.3.4** `static void decaf::lang::System::arraycopy ( const int * src, std::size_t srcPos, int * dest, std::size_t destPos, std::size_t length ) [static]`

Copies the number of elements specified by *length* from the source array starting at the given source offset specified by *srcPos* to the dest array starting at the given destination offset given by *destPos*.

#### Parameters

<i>src</i>	The source array to copy from.
<i>srcPos</i>	The position in the array to start copying from.
<i>dest</i>	The destination array to copy to.
<i>destPos</i>	The position in the destination array to start writing at.
<i>length</i>	The number of elements to copy from <i>src</i> to <i>dest</i> .

#### Exceptions

<i>NullPointerException</i>	if <i>src</i> or <i>dest</i> are NULL.
-----------------------------	--

**6.797.3.5** `static int decaf::lang::System::availableProcessors ( ) [static]`

Returns the number of processors available for execution of Decaf Threads.

This value may change during a particular execution of a Decaf based application. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

#### Returns

the number of available processors.

**6.797.3.6** `static std::string decaf::lang::System::clearProperty ( const std::string & key ) [static]`

Clear any value associated with the system property specified.

#### Parameters

<i>key</i>	The key name of the system property to clear.
------------	---

#### Returns

the previous value of the property named by *key* if there was one, otherwise returns an empty string.

#### Exceptions

<i>IllegalArgumentException</i>	if <i>key</i> is an empty string.
---------------------------------	-----------------------------------

6.797.3.7 `static long long decaf::lang::System::currentTimeMillis ( ) [static]`

Returns the current time in milliseconds.

Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class `Date` for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

#### Returns

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

6.797.3.8 `static const util::Map<std::string, std::string>& decaf::lang::System::getenv ( ) [static]`

Enumerates the system environment and returns a map of env variable names to the string values they hold.

#### Returns

A Map of all environment variables.

#### Exceptions

<b>Exception</b> (p. 1794)	if an error occurs while getting the Environment Map.
----------------------------	---

6.797.3.9 `static std::string decaf::lang::System::getenv ( const std::string & name ) [static]`

Reads an environment value from the system and returns it as a string object.

#### Parameters

<i>name</i>	The environment variable to read.
-------------	-----------------------------------

#### Returns

a string with the value from the variables or ""

#### Exceptions

<i>an</i>	<b>Exception</b> (p. 1794) if an error occurs while reading the Env.
-----------	--

6.797.3.10 `static decaf::util::Properties& decaf::lang::System::getProperties ( )`  
`[static]`

Gets the Properties object that holds the Properties accessed from calls to `getProperty` and `setProperty`.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

### Returns

a reference to the static system Properties object.

6.797.3.11 `static std::string decaf::lang::System::getProperty ( const std::string & key )`  
`[static]`

Gets the specified **System** (p. 3670) property if set, otherwise returns an empty string.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

### Parameters

<i>key</i>	The key name of the desired system property to retrieve.
------------	--

### Returns

an empty string if the named property is not set, otherwise returns the value.

### Exceptions

<i>IllegalArgumentException</i>	if key is an empty string.
---------------------------------	----------------------------

6.797.3.12 `static std::string decaf::lang::System::getProperty ( const std::string & key, const std::string & defaultValue )` `[static]`

Gets the specified **System** (p. 3670) property if set, otherwise returns the specified default value.

If the Properties has not yet been created or are not yet initialized then they will be on the first call to a Properties accessor.

### Parameters

<i>key</i>	The key name of the desired system property to retrieve.
<i>defaultValue</i>	The default value to return if the key is not set in the <b>System</b> (p. 3670) properties.

## Returns

the value of the named system property or the defaultValue if the property isn't set..

## Exceptions

<i>IllegalArgumentException</i>	if key is an empty string.
---------------------------------	----------------------------

### 6.797.3.13 static long long decaf::lang::System::nanoTime ( ) [static]

Returns the current value of the most precise available system timer, in nanoseconds.

This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some code takes to execute:

```
long long startTime = System::nanoTime() (p. 3676); // ... the code being measured ...  
long long estimatedTime = System::nanoTime() (p. 3676) - startTime;
```

## Returns

The current value of the system timer, in nanoseconds.

### 6.797.3.14 static void decaf::lang::System::setenv ( const std::string & name, const std::string & value ) [static]

Sets the specified system property to the value given.

## Parameters

<i>name</i>	The name of the environment variables to set.
<i>value</i>	The value to assign to name.

## Exceptions

<i>an</i>	<b>Exception</b> (p. 1794) if an error occurs when setting the environment variable.
-----------	--

### 6.797.3.15 static std::string decaf::lang::System::setProperty ( const std::string & key, const std::string & value ) [static]

Sets the **System** (p. 3670) Property to the specified value.

**Parameters**

<i>key</i>	The key name of the system property to set to the given value.
<i>value</i>	The value to assign to the key.

**Returns**

the previous value of the property named by key if there was one, otherwise returns an empty string.

**Exceptions**

<i>IllegalArgumentException</i>	if key is an empty string.
---------------------------------	----------------------------

6.797.3.16 `static void decaf::lang::System::unsetenv ( const std::string & name )`  
`[static]`

Clears a set environment value if one is set.

**Parameters**

<i>name</i>	The environment variables to clear.
-------------	-------------------------------------

**Exceptions**

<i>an</i>	<b>Exception</b> (p. 1794) if an error occurs while reading the environment.
-----------	--

**6.797.4 Friends And Related Function Documentation**

6.797.4.1 `friend class decaf::lang::Runtime` `[friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/System.h`

**6.798 activemq::threads::Task Class Reference**

Represents a unit of work that requires one or more iterations to complete.

```
#include <src/main/activemq/threads/Task.h>
```

Inheritance diagram for `activemq::threads::Task`:



## Public Member Functions

- virtual `~Task` ()
- virtual bool `iterate` ()=0

*Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.*

### 6.798.1 Detailed Description

Represents a unit of work that requires one or more iterations to complete.

#### Since

3.0

### 6.798.2 Constructor & Destructor Documentation

6.798.2.1 virtual `activemq::threads::Task::~Task` ( ) [`inline`, `virtual`]

### 6.798.3 Member Function Documentation

6.798.3.1 virtual bool `activemq::threads::Task::iterate` ( ) [`pure virtual`]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

#### Returns

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in `activemq::core::ActiveMQSessionExecutor` (p. 506), `activemq::threads::CompositeTaskRunner` (p. 1195), `activemq::transport::failover::BackupTransportPool` (p. 722), `activemq::transport::failover::CloseTransports` (p. 1123), and `activemq::transport::failover::FailoverTransport` (p. 1841).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Task.h`

## 6.799 decaf::util::concurrent::TaskListener Class Reference

```
#include <src/main/decaf/util/concurrent/TaskListener.h>
```

## Public Member Functions

- virtual `~TaskListener` ()

- virtual void **onTaskComplete** (**lang::Runnable** \*task)=0

*Called when a queued task has completed, the task that finished is passed along for user consumption.*

- virtual void **onTaskException** (**lang::Runnable** \*task, **lang::Exception** &ex)=0

*Called when a queued task has thrown an exception while being run.*

## 6.799.1 Constructor & Destructor Documentation

6.799.1.1 virtual **decaf::util::concurrent::TaskListener::~~TaskListener** ( ) [inline, virtual]

## 6.799.2 Member Function Documentation

6.799.2.1 virtual void **decaf::util::concurrent::TaskListener::onTaskComplete** (**lang::Runnable** \* task ) [pure virtual]

Called when a queued task has completed, the task that finished is passed along for user consumption.

### Parameters

<i>task</i>	Runnable Pointer to the task that finished
-------------	--

6.799.2.2 virtual void **decaf::util::concurrent::TaskListener::onTaskException** (**lang::Runnable** \* task, **lang::Exception** & ex ) [pure virtual]

Called when a queued task has thrown an exception while being run.

The Callee should assume that this was an unrecoverable exeption and that this task is now defunct.

### Parameters

<i>task</i>	Runnable Pointer to the task
<i>ex</i>	The ActiveMQException that was thrown.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TaskListener.h**

## 6.800 activemq::threads::TaskRunner Class Reference

```
#include <src/main/activemq/threads/TaskRunner.h>
```

Inheritance diagram for activemq::threads::TaskRunner:

## Public Member Functions

- virtual `~TaskRunner()`
- virtual void **shutdown** (unsigned int timeout)=0  
*Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.*
- virtual void **shutdown** ()=0  
*Shutdown once the task has finished and the TaskRunner's thread has exited.*
- virtual void **wakeup** ()=0  
*Signal the **TaskRunner** (p. 3680) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3678) instance will be run until its iterate method has returned false indicating it is done.*

## 6.800.1 Constructor & Destructor Documentation

6.800.1.1 virtual activemq::threads::TaskRunner::~TaskRunner ( ) [inline, virtual]

## 6.800.2 Member Function Documentation

6.800.2.1 virtual void activemq::threads::TaskRunner::shutdown ( unsigned int *timeout* ) [pure virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

### Parameters

<i>timeout</i>	- Time in Milliseconds to wait for the task to stop.
----------------	--

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1196), and **activemq::threads::DedicatedTaskRunner** (p. 1639).

6.800.2.2 virtual void activemq::threads::TaskRunner::shutdown ( ) [pure virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1196), and **activemq::threads::DedicatedTaskRunner** (p. 1639).

### 6.800.2.3 virtual void activemq::threads::TaskRunner::wakeup ( ) [pure virtual]

Signal the **TaskRunner** (p. 3680) to wakeup and execute another iteration cycle on the task, the **Task** (p. 3678) instance will be run until its iterate method has returned false indicating it is done.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 1196), and **activemq::threads::DedicatedTaskRunner** (p. 1640).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**TaskRunner.h**

## 6.801 decaf::internal::net::tcp::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

```
#include <src/main/decaf/internal/net/tcp/TcpSocket.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocket:

### Public Member Functions

- **TcpSocket** ( ) throw ( decaf::net::SocketException )  
*Construct a non-connected socket.*
- virtual ~**TcpSocket** ( )  
*Releases the socket handle but not gracefully shut down the connection.*
- SocketHandle **getSocketHandle** ( )  
*Gets the handle for the socket.*
- bool **isConnected** ( ) const
- bool **isClosed** ( ) const
- virtual std::string **getLocalAddress** ( ) const  
*Gets the value of the local Inet address the **Socket** (p. 3445) is bound to if bound, otherwise return the **InetAddress** (p. 1974) ANY value "0.0.0.0".*  
**Returns**  
*the local address bound to, or ANY.*
- virtual void **create** ( ) throw ( decaf::io::IOException )  
*Creates the underlying platform **Socket** (p. 3445) data structures which allows for **Socket** (p. 3445) options to be applied.*  
*The created socket is in an unconnected state.*  
**Exceptions**

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **accept** (SocketImpl \*socket) throw ( decaf::io::IOException )

- virtual void **bind** (const std::string &ipaddress, int **port**) throw ( decaf::io::IOException )

*Binds this **Socket** (p. 3445) instance to the local ip address and port number given.*

#### Parameters

ipaddress	The address of local ip to bind to.
port	The port number on the host to bind to.

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **connect** (const std::string &hostname, int **port**, int timeout) throw ( decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException )

*Connects this socket to the given host and port.*

#### Parameters

hostname	The name of the host to connect to, or IP address.
port	The port number on the host to connect to.
timeout	Time in milliseconds to wait for a connection, 0 indicates forever.

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
<b>SocketTimeoutException</b> (p. 3487)	if the connect call times out due to timeout being set.
IllegalArgumentException	if a parameter has an illegal value.

- virtual void **listen** (int backlog) throw ( decaf::io::IOException )

*Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.*

*If a connection indication arrives when the queue is full, the connection is refused.*

#### Parameters

backlog	The maximum length of the connection queue.
---------	---

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual **decaf::io::InputStream \* getInputStream** () throw ( decaf::io::IOException )

*Gets the InputStream linked to this **Socket** (p. 3445).*

#### Returns

*an InputStream pointer owned by the **Socket** (p. 3445) object.*

#### Exceptions

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual **decaf::io::OutputStream \* getOutputStream** () throw ( decaf::io::IOException )

*Gets the OutputStream linked to this **Socket** (p. 3445).*

**Returns**

an *OutputStream* pointer owned by the **Socket** (p. 3445) object.

**Exceptions**

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual int **available** () throw ( decaf::io::IOException )

*Gets the number of bytes that can be read from the **Socket** (p. 3445) without blocking.*

**Returns**

*the number of bytes that can be read from the **Socket** (p. 3445) without blocking.*

**Exceptions**

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **close** () throw ( decaf::io::IOException )

*Closes the socket, terminating any blocked reads or writes.*

**Exceptions**

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **shutdownInput** () throw ( decaf::io::IOException )

*Places the input stream for this socket at "end of stream".*

*Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3480) on the socket, the stream will return EOF.*

**Exceptions**

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual void **shutdownOutput** () throw ( decaf::io::IOException )

*Disables the output stream for this socket.*

*For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3480) on the socket, the stream will throw an IOException.*

**Exceptions**

IOException	if an I/O error occurs while attempting this operation.
-------------	---

- virtual int **getOption** (int option) const throw ( decaf::io::IOException )

*Gets the specified **Socket** (p. 3445) option.*

**Parameters**

option	The <b>Socket</b> (p. 3445) options whose value is to be retrieved.
--------	---

**Returns**

*the value of the given socket option.*

**Exceptions**

IOException	if an I/O error occurs while performing this operation.
-------------	---

- virtual void **setOption** (int option, int value) throw ( decaf::io::IOException )

*Sets the specified option on the **Socket** (p. 3445) if supported.*

**Parameters**

option	The <b>Socket</b> (p. 3445) option to set.
--------	--

value	The value of the socket option to apply to the socket.
-------	--

**Exceptions**

IOException	if an I/O error occurs while performing this operation.
-------------	---

- int **read** (unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Reads the requested data from the Socket and write it into the passed in buffer.*

- void **write** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

*Writes the specified data in the passed in buffer to the Socket.*

**Protected Member Functions**

- void **checkResult** (apr\_status\_t value) const throw ( decaf::net::SocketException )

**6.801.1 Detailed Description**

Platform-independent implementation of the socket interface.

**6.801.2 Constructor & Destructor Documentation**

- 6.801.2.1 decaf::internal::net::tcp::TcpSocket::TcpSocket ( ) throw ( decaf::net::SocketException )

Construct a non-connected socket.

**Exceptions**

SocketException	thrown if an error occurs while creating the Socket.
-----------------	--

- 6.801.2.2 virtual decaf::internal::net::tcp::TcpSocket::~TcpSocket ( ) [virtual]

Releases the socket handle but not gracefully shut down the connection.

**6.801.3 Member Function Documentation**

- 6.801.3.1 virtual void decaf::internal::net::tcp::TcpSocket::accept ( SocketImpl \* socket ) throw ( decaf::io::IOException ) [virtual]

6.801.3.2 `virtual int decaf::internal::net::tcp::TcpSocket::available ( ) throw ( decaf::io::IOException ) [virtual]`

Gets the number of bytes that can be read from the **Socket** (p. 3445) without blocking.

#### Returns

the number of bytes that can be read from the **Socket** (p. 3445) without blocking.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3475).

6.801.3.3 `virtual void decaf::internal::net::tcp::TcpSocket::bind ( const std::string & ipaddress, int port ) throw ( decaf::io::IOException ) [virtual]`

Binds this **Socket** (p. 3445) instance to the local ip address and port number given.

#### Parameters

<i>ipaddress</i>	The address of local ip to bind to.
<i>port</i>	The port number on the host to bind to.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3475).

6.801.3.4 `void decaf::internal::net::tcp::TcpSocket::checkResult ( apr_status_t value ) const throw ( decaf::net::SocketException ) [protected]`

6.801.3.5 `virtual void decaf::internal::net::tcp::TcpSocket::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes the socket, terminating any blocked reads or writes.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3475).



6.801.3.6 virtual void decaf::internal::net::tcp::TcpSocket::connect ( const std::string & *hostname*, int *port*, int *timeout* ) throw ( decaf::io::IOException, decaf::net::SocketException, decaf::lang::exceptions::IllegalArgumentException ) [virtual]

Connects this socket to the given host and port.

#### Parameters

<i>hostname</i>	The name of the host to connect to, or IP address.
<i>port</i>	The port number on the host to connect to.
<i>timeout</i>	Time in milliseconds to wait for a connection, 0 indicates forever.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
<b>SocketTimeoutException</b> (p. 3487)	if the connect call times out due to timeout being set.
<i>IllegalArgumentEx-ception</i>	if a parameter has an illegal value.

Implements **decaf::net::SocketImpl** (p. 3476).

6.801.3.7 virtual void decaf::internal::net::tcp::TcpSocket::create ( ) throw ( decaf::io::IOException ) [virtual]

Creates the underlying platform **Socket** (p. 3445) data structures which allows for **Socket** (p. 3445) options to be applied.

The created socket is in an unconnected state.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3476).

6.801.3.8 virtual decaf::io::InputStream\* decaf::internal::net::tcp::TcpSocket::getInputStream ( ) throw ( decaf::io::IOException ) [virtual]

Gets the InputStream linked to this **Socket** (p. 3445).

#### Returns

an InputStream pointer owned by the **Socket** (p. 3445) object.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3477).

6.801.3.9 `virtual std::string decaf::internal::net::tcp::TcpSocket::getLocalAddress ( ) const`  
[virtual]

Gets the value of the local Inet address the **Socket** (p.3445) is bound to if bound, otherwise return the **InetAddress** (p. 1974) ANY value "0.0.0.0".

#### Returns

the local address bound to, or ANY.

Implements **decaf::net::SocketImpl** (p. 3477).

6.801.3.10 `virtual int decaf::internal::net::tcp::TcpSocket::getOption ( int option ) const throw ( decaf::io::IOException )` [virtual]

Gets the specified **Socket** (p. 3445) option.

#### Parameters

<i>option</i>	The <b>Socket</b> (p. 3445) options whose value is to be retrieved.
---------------	---

#### Returns

the value of the given socket option.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3478).

6.801.3.11 `virtual decaf::io::OutputStream* decaf::internal::net::tcp::TcpSocket::getOutputStream ( ) throw ( decaf::io::IOException )` [virtual]

Gets the OutputStream linked to this **Socket** (p. 3445).

#### Returns

an OutputStream pointer owned by the **Socket** (p. 3445) object.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3478).

6.801.3.12 `SocketHandle decaf::internal::net::tcp::TcpSocket::getSocketHandle ( )`  
[inline]

Gets the handle for the socket.

#### Returns

SocketHabler for this Socket, can be NULL

6.801.3.13 `bool decaf::internal::net::tcp::TcpSocket::isClosed ( ) const` [inline]

#### Returns

true if the close method has been called on this Socket.

6.801.3.14 `bool decaf::internal::net::tcp::TcpSocket::isConnected ( ) const` [inline]

#### Returns

true if the socketHandle is not in a disconnected state.

6.801.3.15 `virtual void decaf::internal::net::tcp::TcpSocket::listen ( int backlog ) throw ( decaf::io::IOException )` [virtual]

Sets the maximum queue length for incoming connection indications (a request to connect) to the count argument.

If a connection indication arrives when the queue is full, the connection is refused.

#### Parameters

<i>backlog</i>	The maximum length of the connection queue.
----------------	---

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3479).

6.801.3.16 `int decaf::internal::net::tcp::TcpSocket::read ( unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )`

Reads the requested data from the Socket and write it into the passed in buffer.

**Parameters**

<i>buffer</i>	The buffer to read into
<i>size</i>	The size of the specified buffer
<i>offset</i>	The offset into the buffer where reading should start filling.
<i>length</i>	The number of bytes past offset to fill with data.

**Returns**

the actual number of bytes read or -1 if at EOF.

**Exceptions**

<i>IOException</i>	if an I/O error occurs during the read.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

6.801.3.17 `virtual void decaf::internal::net::tcp::TcpSocket::setOption ( int option, int value )  
throw ( decaf::io::IOException ) [virtual]`

Sets the specified option on the **Socket** (p. 3445) if supported.

**Parameters**

<i>option</i>	The <b>Socket</b> (p. 3445) option to set.
<i>value</i>	The value of the socket option to apply to the socket.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while performing this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3479).

6.801.3.18 `virtual void decaf::internal::net::tcp::TcpSocket::shutdownInput ( ) throw ( decaf::io::IOException ) [virtual]`

Places the input stream for this socket at "end of stream".

Any data sent to this socket is acknowledged and then silently discarded. If you read from a socket input stream after invoking **shutdownInput()** (p. 3480) on the socket, the stream will return EOF.

**Exceptions**

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3480).

6.801.3.19 virtual void decaf::internal::net::tcp::TcpSocket::shutdownOutput ( ) throw ( decaf::io::IOException ) [virtual]

Disables the output stream for this socket.

For a TCP socket, any previously written data will be sent followed by TCP's normal connection termination sequence. If you write to a socket output stream after invoking **shutdownOutput()** (p. 3480) on the socket, the stream will throw an IOException.

#### Exceptions

<i>IOException</i>	if an I/O error occurs while attempting this operation.
--------------------	---

Implements **decaf::net::SocketImpl** (p. 3480).

6.801.3.20 void decaf::internal::net::tcp::TcpSocket::write ( const unsigned char \* *buffer*, int *size*, int *offset*, int *length* ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::NullPointerException )

Writes the specified data in the passed in buffer to the Socket.

#### Parameters

<i>buffer</i>	The buffer to write to the socket.
<i>size</i>	The size of the specified buffer.
<i>offset</i>	The offset into the buffer where the data to write starts at.
<i>length</i>	The number of bytes past offset to write.

#### Exceptions

<i>IOException</i>	if an I/O error occurs during the write.
<i>NullPointerException</i>	if buffer is Null.
<i>IndexOutOfBoundsException</i>	if offset + length is greater than buffer size.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/tcp/**TcpSocket.h**

## 6.802 decaf::internal::net::tcp::TcpSocketInputStream Class Reference

Input stream for performing reads on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketInputStream.h>
```

Inheritance diagram for `decaf::internal::net::tcp::TcpSocketInputStream`:

## Public Member Functions

- **TcpSocketInputStream** (**TcpSocket** \*socket)

*Create a new `InputStream` to use for reading from the TCP/IP socket.*

- virtual **~TcpSocketInputStream** ()
- virtual int **available** () const throw ( `decaf::io::IOException` )

*Indicates the number of bytes available.*

*The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute. The default implementation of this method returns zero.*

### Returns

*the number of bytes available on this input stream.*

### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
------------------------------	-------------------------

- virtual void **close** () throw ( `decaf::io::IOException` )

*Close - does nothing.*

- virtual long long **skip** (long long num) throw ( `decaf::io::IOException`, `decaf::lang::exceptions::UnsupportedOperationException` )

*Not supported.*

## Protected Member Functions

- virtual int **doReadByte** () throw ( `io::IOException` )
- virtual int **doReadArrayBounded** (unsigned char \*buffer, int size, int offset, int length) throw ( `decaf::io::IOException`, `decaf::lang::exceptions::IndexOutOfBoundsException`, `decaf::lang::exceptions::NullPointerException` )

### 6.802.1 Detailed Description

Input stream for performing reads on a socket.

This class will only work properly for blocking sockets.

### Since

1.0

### 6.802.2 Constructor & Destructor Documentation

6.802.2.1 `decaf::internal::net::tcp::TcpSocketInputStream::TcpSocketInputStream (   
 TcpSocket * socket )`

Create a new `InputStream` to use for reading from the TCP/IP socket.

#### Parameters

<code>socket</code>	The parent <code>SocketImpl</code> for this stream.
---------------------	---

6.802.2.2 `virtual decaf::internal::net::tcp::TcpSocketInputStream::~~TcpSocketInputStream ( )   
 [virtual]`

#### 6.802.3 Member Function Documentation

6.802.3.1 `virtual int decaf::internal::net::tcp::TcpSocketInputStream::available ( ) const throw (   
 decaf::io::IOException ) [virtual]`

Indicates the number of bytes available.

The default implementation of this methods returns 0. Classes that override this method may return the total number of bytes that are currently available for reading and others may simply return a value of one indicating that there is some data available. The caller should view the result of this method as an absolute.

The default implementation of this method returns zero.

#### Returns

the number of bytes available on this input stream.

#### Exceptions

<b><code>IOException</code></b> (p. 2103)	if an I/O error occurs.
--	-------------------------

Reimplemented from `decaf::io::InputStream` (p. 2004).

6.802.3.2 `virtual void decaf::internal::net::tcp::TcpSocketInputStream::close ( ) throw (   
 decaf::io::IOException ) [virtual]`

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Closes the **InputStream** (p. 2002) freeing any resources that might have been acquired during the lifetime of this stream.

The default implementation of this method does nothing.

Reimplemented from `decaf::io::InputStream` (p. 2004).

```
6.802.3.3  virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadArrayBounded
( unsigned char * buffer, int size, int offset,
  int length ) throw ( decaf::io::IOException,
    decaf::lang::exceptions::IndexOutOfBoundsException,
    decaf::lang::exceptions::NullPointerException ) [protected,
    virtual]
```

Reimplemented from **decaf::io::InputStream** (p. 2005).

```
6.802.3.4  virtual int decaf::internal::net::tcp::TcpSocketInputStream::doReadByte ( ) throw (
    io::IOException ) [protected, virtual]
```

Implements **decaf::io::InputStream** (p. 2005).

```
6.802.3.5  virtual long long decaf::internal::net::tcp::TcpSocketInputStream::skip
( long long num ) throw ( decaf::io::IOException,
    decaf::lang::exceptions::UnsupportedOperationException )
[virtual]
```

Not supported.

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.

The skip method of **InputStream** (p. 2002) creates a byte array and then repeatedly reads into it until *num* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

#### Parameters

<i>num</i>	The number of bytes to skip.
------------	------------------------------

#### Returns

total bytes skipped

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<i>UnsupportedOperationException</i>	if the concrete stream class does not support skipping bytes.

Reimplemented from **decaf::io::InputStream** (p. 2010).

The documentation for this class was generated from the following file:



- src/main/decaf/internal/net/tcp/TcpSocketInputStream.h

## 6.803 decaf::internal::net::tcp::TcpSocketOutputStream Class Reference

Output stream for performing write operations on a socket.

```
#include <src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h>
```

Inheritance diagram for decaf::internal::net::tcp::TcpSocketOutputStream:

### Public Member Functions

- **TcpSocketOutputStream** (**TcpSocket** \*socket)  
*Create a new instance of a Socket OutputStream class.*
- virtual ~**TcpSocketOutputStream** ()
- virtual void **close** () throw ( decaf::io::IOException )  
*Closes this object and deallocates the appropriate resources.  
The object is generally no longer usable after calling close.*

#### Exceptions

<b>IOException</b> (p. 2103)	<i>if an error occurs while closing.</i>
------------------------------	--

*The default implementation of this method does nothing.*

### Protected Member Functions

- virtual void **doWriteByte** (unsigned char c) throw ( decaf::io::IOException )
- virtual void **doWriteArrayBounded** (const unsigned char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )

#### 6.803.1 Detailed Description

Output stream for performing write operations on a socket.

#### Since

1.0

#### 6.803.2 Constructor & Destructor Documentation

6.803.2.1 `decaf::internal::net::tcp::TcpSocketOutputStream::TcpSocketOutputStream ( TcpSocket * socket )`

Create a new instance of a Socket OutputStream class.

#### Parameters

<i>socket</i>	The socket to use to write out the data.
---------------	--

6.803.2.2 `virtual decaf::internal::net::tcp::TcpSocketOutputStream::~~TcpSocketOutputStream ( ) [virtual]`

### 6.803.3 Member Function Documentation

6.803.3.1 `virtual void decaf::internal::net::tcp::TcpSocketOutputStream::close ( ) throw ( decaf::io::IOException ) [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	if an error occurs while closing.
--	-----------------------------------

The default implementation of this method does nothing.

The default implementation of this method does nothing.

Reimplemented from **decaf::io::OutputStream** (p. 2858).

6.803.3.2 `virtual void decaf::internal::net::tcp::TcpSocketOutputStream::doWriteArrayBounded ( const unsigned char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException ) [protected, virtual]`

Reimplemented from **decaf::io::OutputStream** (p. 2859).

6.803.3.3 `virtual void decaf::internal::net::tcp::TcpSocketOutputStream::doWriteByte ( unsigned char c ) throw ( decaf::io::IOException ) [protected, virtual]`

Implements **decaf::io::OutputStream** (p. 2859).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h`

## 6.804 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2105).

```
#include <src/main/activemq/transport/tcp/TcpTransport.h>
```

Inheritance diagram for activemq::transport::tcp::TcpTransport:

### Public Member Functions

- **TcpTransport** (const **Pointer**< **Transport** > &next)  
*Creates a new instance of a **TcpTransport** (p. 3696), the transport is left unconnected and is in a unusable state until the connect method is called.*
- virtual ~**TcpTransport** ()
- void **connect** (const **decaf::net::URI** &uri, const **decaf::util::Properties** &properties)  
*Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.*
- virtual void **close** () throw ( **decaf::io::IOException** )  
*Delegates to the superclass and then closes the socket.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3819) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3819) been shutdown and no longer usable.*

### Protected Member Functions

- virtual **decaf::net::Socket** \* **createSocket** ()  
*Create an unconnected Socket instance to be used by the transport to communicate with the broker.*
- virtual void **configureSocket** (**decaf::net::Socket** \*socket, const **decaf::util::Properties** &properties)  
*Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.*

#### 6.804.1 Detailed Description

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2105).

The lower level transport should take care of managing stream reads and writes.

## 6.804.2 Constructor & Destructor Documentation

6.804.2.1 `activemq::transport::tcp::TcpTransport::TcpTransport ( const Pointer< Transport > & next )`

Creates a new instance of a **TcpTransport** (p. 3696), the transport is left unconnected and is in a unusable state until the connect method is called.

### Parameters

<i>next</i>	The next transport in the chain
-------------	---------------------------------

6.804.2.2 `virtual activemq::transport::tcp::TcpTransport::~~TcpTransport ( ) [virtual]`

## 6.804.3 Member Function Documentation

6.804.3.1 `virtual void activemq::transport::tcp::TcpTransport::close ( ) throw ( decaf::io::IOException ) [virtual]`

Delegates to the superclass and then closes the socket.

### Exceptions

<i>IOException</i>	if errors occur.
--------------------	------------------

Reimplemented from **activemq::transport::TransportFilter** (p. 3829).

6.804.3.2 `virtual void activemq::transport::tcp::TcpTransport::configureSocket ( decaf::net::Socket * socket, const decaf::util::Properties & properties ) [protected, virtual]`

Using options from configuration URI, configure the socket options before the Socket instance is connected to the Server.

Subclasses can override this option to set more configuration options, they should called the base class version to allow the default set of Socket options to also be configured.

### Parameters

<i>socket</i>	The Socket instance to configure using options from the given Properties.
---------------	---

### Exceptions

<i>NullPointerException</i>	if the Socket instance is null.
<i>IllegalArgumentException</i>	if the socket instance is not handled by the class.
<i>SocketException</i>	if there is an error while setting one of the Socket options.

6.804.3.3 `void activemq::transport::tcp::TcpTransport::connect ( const decaf::net::URI & uri, const decaf::util::Properties & properties )`

Creates a Socket and configures it before attempting to connect to the location specified by the URI passed in.

The Socket is configured using parameters in the properties that are passed to this method.

#### Parameters

<i>uri</i>	The URI that the <b>Transport</b> (p. 3819) is to connect to once initialized.
<i>properties</i>	The Properties that have been parsed from the URI or from configuration files.

6.804.3.4 `virtual decaf::net::Socket* activemq::transport::tcp::TcpTransport::createSocket ( ) [protected, virtual]`

Create an unconnected Socket instance to be used by the transport to communicate with the broker.

#### Returns

a newly created unconnected Socket instance.

#### Exceptions

<i>IOException</i>	if there is an error while creating the unconnected Socket.
--------------------	---

Reimplemented in **activemq::transport::tcp::SslTransport** (p. 3519).

6.804.3.5 `virtual bool activemq::transport::tcp::TcpTransport::isClosed ( ) const [inline, virtual]`

Has the **Transport** (p. 3819) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3819)

Reimplemented from **activemq::transport::TransportFilter** (p. 3830).

6.804.3.6 `virtual bool activemq::transport::tcp::TcpTransport::isConnected ( ) const [inline, virtual]`

Is the **Transport** (p. 3819) Connected to its Broker.

#### Returns

true if a connection has been made.

Reimplemented from **activemq::transport::TransportFilter** (p. 3831).

6.804.3.7 `virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant ( ) const`  
`[inline, virtual]`

Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.

### Returns

true if the **Transport** (p. 3819) is fault tolerant.

Reimplemented from **activemq::transport::TransportFilter** (p. 3831).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransport.h`

## 6.805 activemq::transport::tcp::TcpTransportFactory Class Reference

Factory Responsible for creating the **TcpTransport** (p. 3696).

```
#include <src/main/activemq/transport/tcp/TcpTransportFactory.h>
```

Inheritance diagram for `activemq::transport::tcp::TcpTransportFactory`:

### Public Member Functions

- virtual `~TcpTransportFactory ()`
- virtual `Pointer< Transport > create (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`  
*Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.*
- virtual `Pointer< Transport > createComposite (const decaf::net::URI &location) throw ( exceptions::ActiveMQException )`  
*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

### Protected Member Functions

- virtual `Pointer< Transport > doCreateComposite (const decaf::net::URI &location, const Pointer< wireformat::WireFormat > &wireFormat, const decaf::util::Properties &properties) throw ( exceptions::ActiveMQException )`  
*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

### 6.805.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 3696).

### 6.805.2 Constructor & Destructor Documentation

6.805.2.1 `virtual activemq::transport::tcp::TcpTransportFactory::~~TcpTransportFactory ( )`  
`[inline, virtual]`

### 6.805.3 Member Function Documentation

6.805.3.1 `virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::create`  
`( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException`  
`) [virtual]`

Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 3826).

6.805.3.2 `virtual Pointer<Transport> ac-`  
`tivemq::transport::tcp::TcpTransportFactory::createComposite (`  
`const decaf::net::URI & location ) throw ( exceptions::ActiveMQException )`  
`[virtual]`

Creates a slimed down **Transport** (p. 3819) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implements **activemq::transport::TransportFactory** (p. 3827).

```

6.805.3.3 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::doCreateComposite
( const decaf::net::URI & location, const Pointer<WireFormat>
  & wireFormat, const decaf::util::Properties & properties ) throw (
  exceptions::ActiveMQException ) [protected, virtual]

```

Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to.
<i>wireFormat</i>	- the assigned WireFormat for the new <b>Transport</b> (p. 3819).
<i>properties</i>	- Properties to apply to the transport.

#### Returns

new Pointer to a **TcpTransport** (p. 3696).

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Reimplemented in **activemq::transport::tcp::SslTransportFactory** (p. 3520).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransportFactory.h**

## 6.806 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 3093) based **Destination** (p. 1688).

```
#include <src/main/cms/TemporaryQueue.h>
```

Inheritance diagram for cms::TemporaryQueue:

#### Public Member Functions

- virtual **~TemporaryQueue** ()
- virtual std::string **getQueueName** () const =0 throw ( CMSEException )  
*Gets the name of this queue.*
- virtual void **destroy** ()=0 throw ( CMSEException )  
*Destroy's the Temporary **Destination** (p. 1688) at the Provider.*



### 6.806.1 Detailed Description

Defines a Temporary **Queue** (p. 3093) based **Destination** (p. 1688).

A **TemporaryQueue** (p. 3701) is a special type of **Queue** (p. 3093) **Destination** (p. 1688) that can only be consumed from the **Connection** (p. 1232) which created it. TemporaryQueues are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryQueue** (p. 3701) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1232) that created it.

#### Since

1.0

### 6.806.2 Constructor & Destructor Documentation

6.806.2.1 `virtual cms::TemporaryQueue::~TemporaryQueue ( ) [inline, virtual]`

### 6.806.3 Member Function Documentation

6.806.3.1 `virtual void cms::TemporaryQueue::destroy ( ) throw ( CMSEException ) [pure virtual]`

Destroy's the Temporary **Destination** (p. 1688) at the Provider.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 576).

6.806.3.2 `virtual std::string cms::TemporaryQueue::getQueueName ( ) const throw ( CMSEException ) [pure virtual]`

Gets the name of this queue.

#### Returns

The queue name.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
--	--------------------------------

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 578).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryQueue.h`

## 6.807 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 3757) based **Destination** (p. 1688).

```
#include <src/main/cms/TemporaryTopic.h>
```

Inheritance diagram for cms::TemporaryTopic:

### Public Member Functions

- virtual `~TemporaryTopic()`
- virtual `std::string getTopicName()` `const =0` throw ( `CMSEException` )  
*Gets the name of this topic.*
- virtual void `destroy()` `=0` throw ( `CMSEException` )  
*Destroy's the Temporary **Destination** (p. 1688) at the Provider.*

#### 6.807.1 Detailed Description

Defines a Temporary **Topic** (p. 3757) based **Destination** (p. 1688).

A **TemporaryTopic** (p. 3703) is a special type of **Topic** (p. 3757) **Destination** (p. 1688) that can only be consumed from the **Connection** (p. 1232) which created it. Temporary-Topics are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryTopic** (p. 3703) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1232) that created it.

#### Since

1.0

#### 6.807.2 Constructor & Destructor Documentation

6.807.2.1 `virtual cms::TemporaryTopic::~TemporaryTopic( )` `[inline, virtual]`

#### 6.807.3 Member Function Documentation

6.807.3.1 `virtual void cms::TemporaryTopic::destroy( )` throw ( `CMSEException` ) `[pure virtual]`

Destroy's the Temporary **Destination** (p. 1688) at the Provider.

**Exceptions**

<b>CMSEException</b> (p. 1130)	
-----------------------------------	--

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 604).

6.807.3.2 `virtual std::string cms::TemporaryTopic::getTopicName ( ) const throw ( CMSEException ) [pure virtual]`

Gets the name of this topic.

**Returns**

The topic name.

**Exceptions**

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 606).

The documentation for this class was generated from the following file:

- src/main/cms/**TemporaryTopic.h**

**6.808 cms::TextMessage Class Reference**

Interface for a text message.

```
#include <src/main/cms/TextMessage.h>
```

Inheritance diagram for cms::TextMessage:

**Public Member Functions**

- virtual `~TextMessage ()`
- virtual `std::string getText () const =0 throw ( cms::CMSEException )`  
*Gets the message character buffer.*
- virtual `void setText (const char *msg)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )`  
*Sets the message contents, does not take ownership of the passed char\*, but copies it instead.*
- virtual `void setText (const std::string &msg)=0 throw ( cms::MessageNotWriteableException, cms::CMSEException )`  
*Sets the message contents.*

### 6.808.1 Detailed Description

Interface for a text message.

A **TextMessage** (p. 3704) can contain any Text based pay load such as an XML Document or other Text based document.

Like all Messages, a **TextMessage** (p. 3704) is received in Read-Only mode, any attempt to write to the **Message** (p. 2493) will result in a `MessageNotWritableException` being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

#### Since

1.0

### 6.808.2 Constructor & Destructor Documentation

6.808.2.1 `virtual cms::TextMessage::~TextMessage ( ) [inline, virtual]`

### 6.808.3 Member Function Documentation

6.808.3.1 `virtual std::string cms::TextMessage::getText ( ) const throw ( cms::CMSEException ) [pure virtual]`

Gets the message character buffer.

#### Returns

The message character buffer.

#### Exceptions

<b>CMSEException</b> (p. 1130)	- if an internal error occurs.
-----------------------------------	--------------------------------

Implemented in `activemq::commands::ActiveMQTextMessage` (p. 634).

6.808.3.2 `virtual void cms::TextMessage::setText ( const std::string & msg ) throw ( cms::MessageNotWriteableException, cms::CMSEException ) [pure virtual]`

Sets the message contents.

#### Parameters

<i>msg</i>	The message buffer.
------------	---------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 634).

```
6.808.3.3 virtual void cms::TextMessage::setText ( const char * msg ) throw (
    cms::MessageNotWriteableException, cms::CMSEException ) [pure
    virtual]
```

Sets the message contents, does not take ownership of the passed char\*, but copies it instead.

**Parameters**

<i>msg</i>	The message buffer.
------------	---------------------

**Exceptions**

<b><i>CMSEException</i></b> (p. 1130)	- if an internal error occurs.
<b><i>MessageNotWriteableException</i></b> (p. 2680)	- if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 635).

The documentation for this class was generated from the following file:

- src/main/cms/**TextMessage.h**

**6.809 decaf::lang::Thread Class Reference**

A **Thread** (p. 3707) is a concurrent unit of execution.

```
#include <src/main/decaf/lang/Thread.h>
```

Inheritance diagram for decaf::lang::Thread:

**Data Structures**

- class **UncaughtExceptionHandler**

Interface for handlers invoked when a **Thread** (p. 3707) abruptly terminates due to an uncaught exception.

## Public Types

- enum **State** {  
**NEW** = 0, **RUNNABLE** = 1, **BLOCKED** = 2, **WAITING** = 3,  
**TIMED\_WAITING** = 4, **SLEEPING** = 5, **TERMINATED** = 6 }  
Represents the various states that the **Thread** (p. 3707) can be in during its lifetime.

## Public Member Functions

- **Thread** ()  
Constructs a new **Thread** (p. 3707).
- **Thread** (**Runnable** \*task)  
Constructs a new **Thread** (p. 3707) with the given target **Runnable** (p. 3264) task.
- **Thread** (const std::string &name)  
Constructs a new **Thread** (p. 3707) with the given name.
- **Thread** (**Runnable** \*task, const std::string &name)  
Constructs a new **Thread** (p. 3707) with the given target **Runnable** (p. 3264) task and name.
- virtual ~**Thread** ()
- virtual void **start** () throw ( decaf::lang::exceptions::IllegalThreadStateException, decaf::lang::exceptions::RuntimeException )  
Creates a system thread and starts it in a joinable mode.
- virtual void **join** () throw ( decaf::lang::exceptions::InterruptedException )  
Forces the Current **Thread** (p. 3707) to wait until the thread exits.
- virtual void **join** (long long millisecs) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException )  
Forces the Current **Thread** (p. 3707) to wait until the thread exits.
- virtual void **join** (long long millisecs, unsigned int nanos) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException )  
Forces the Current **Thread** (p. 3707) to wait until the thread exits.
- virtual void **run** ()  
Default implementation of the run method - does nothing.
- std::string **getName** () const  
Returns the Thread's assigned name.
- void **setName** (const std::string &name)  
Sets the name of the **Thread** (p. 3707) to the new Name given by the argument name
- int **getPriority** () const  
Gets the currently set priority for this **Thread** (p. 3707).
- void **setPriority** (int value) throw ( decaf::lang::exceptions::IllegalArgumentException )

*Sets the current Thread's priority to the newly specified value.*

- const **UncaughtExceptionHandler** \* **getUncaughtExceptionHandler** () const

*Set the handler invoked when this thread abruptly terminates due to an uncaught exception.*

- void **setUncaughtExceptionHandler** (**UncaughtExceptionHandler** \*handler)

*Set the handler invoked when this thread abruptly terminates due to an uncaught exception.*

- std::string **toString** () const

*Returns a string that describes the **Thread** (p. 3707).*

- bool **isAlive** () const

*Returns true if the **Thread** (p. 3707) is alive, meaning it has been started and has not yet died.*

- **Thread::State** **getState** () const

*Returns the currently set State of this **Thread** (p. 3707).*

## Static Public Member Functions

- static void **sleep** (long long millisecs) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException )

*Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.*

- static void **sleep** (long long millisecs, unsigned int nanos) throw ( lang::exceptions::InterruptedException, lang::exceptions::IllegalArgumentException )

*Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.*

- static void **yield** ()

*Causes the currently executing thread object to temporarily pause and allow other threads to execute.*

- static long long **getId** ()

*Obtains the **Thread** (p. 3707) Id of the current thread.*

- static **Thread** \* **currentThread** ()

*Returns a pointer to the currently executing thread object.*

## Static Public Attributes

- static const int **MIN\_PRIORITY** = 1

*The minimum priority that a thread can have.*

- static const int **NORM\_PRIORITY** = 5

*The default priority that a thread is given at create time.*

- static const int **MAX\_PRIORITY** = 10

*The maximum priority that a thread can have.*

## Friends

- class **decaf::util::concurrent::locks::LockSupport**
- class **decaf::lang::Runtime**

### 6.809.1 Detailed Description

A **Thread** (p. 3707) is a concurrent unit of execution.

It has its own call stack for methods being invoked, their arguments and local variables. Each process has at least one main **Thread** (p. 3707) running when it is started; typically, there are several others for housekeeping. The application might decide to launch additional **Threads** for specific purposes.

**Threads** in the same process interact and synchronize by the use of shared objects and monitors associated with these objects.

There are basically two main ways of having a **Thread** (p. 3707) execute application code. One is providing a new class that extends **Thread** (p. 3707) and overriding its **run()** (p. 3713) method. The other is providing a new **Thread** (p. 3707) instance with a **Runnable** (p. 3264) object during its creation. In both cases, the **start()** (p. 3715) method must be called to actually execute the new **Thread** (p. 3707).

Each **Thread** (p. 3707) has an integer priority that basically determines the amount of CPU time the **Thread** (p. 3707) gets. It can be set using the **setPriority(int)** (p. 3714) method. A **Thread** (p. 3707) can also be made a daemon, which makes it run in the background. The latter also affects VM termination behavior: the VM does not terminate automatically as long as there are non-daemon threads running.

#### See also

**decaf.lang.ThreadGroup** (p. 3717)

#### Since

1.0

### 6.809.2 Member Enumeration Documentation

#### 6.809.2.1 enum **decaf::lang::Thread::State**

Represents the various states that the **Thread** (p. 3707) can be in during its lifetime.

#### Enumerator:

**NEW** Before a **Thread** (p. 3707) is started it exists in this State.

**RUNNABLE** While a **Thread** (p. 3707) is running and is not blocked it is in this State.

**BLOCKED** A **Thread** (p. 3707) that is waiting to acquire a lock is in this state.

**WAITING** A **Thread** (p. 3707) that is waiting for another **Thread** (p. 3707) to perform an action is in this state.



**TIMED\_WAITING** A **Thread** (p. 3707) that is waiting for another **Thread** (p. 3707) to perform an action up to a specified time interval is in this state.

**SLEEPING** A **Thread** (p. 3707) that is blocked in a Sleep call is in this state.

**TERMINATED** A **Thread** (p. 3707) whose run method has exited is in this state.

### 6.809.3 Constructor & Destructor Documentation

#### 6.809.3.1 decaf::lang::Thread::Thread ( )

Constructs a new **Thread** (p. 3707).

This constructor has the same effect as Thread( NULL, NULL, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

#### 6.809.3.2 decaf::lang::Thread::Thread ( Runnable \* task )

Constructs a new **Thread** (p. 3707) with the given target **Runnable** (p. 3264) task.

This constructor has the same effect as Thread( NULL, task, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

##### Parameters

<i>task</i>	the <b>Runnable</b> (p. 3264) that this thread manages, if the task is NULL the Thread's run method is used instead.
-------------	--

#### 6.809.3.3 decaf::lang::Thread::Thread ( const std::string & name )

Constructs a new **Thread** (p. 3707) with the given name.

This constructor has the same effect as Thread( NULL, NULL, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is automatically generated and are of the form "Thread-"+n, where n is an integer.

##### Parameters

<i>name</i>	the name to assign to this <b>Thread</b> (p. 3707).
-------------	---

#### 6.809.3.4 decaf::lang::Thread::Thread ( Runnable \* task, const std::string & name )

Constructs a new **Thread** (p. 3707) with the given target **Runnable** (p. 3264) task and name.

This constructor has the same effect as Thread( NULL, task, GIVEN\_NAME ), where GIVEN\_NAME is a newly generated name. When no name is given the name is auto-

matically generated and are of the form "Thread-"+n, where n is an integer.

#### Parameters

<i>task</i>	the <b>Runnable</b> (p. 3264) that this thread manages, if the task is NULL the Thread's run method is used instead.
<i>name</i>	the name to assign to this <b>Thread</b> (p. 3707).

6.809.3.5 `virtual decaf::lang::Thread::~~Thread ( ) [virtual]`

#### 6.809.4 Member Function Documentation

6.809.4.1 `static Thread* decaf::lang::Thread::currentThread ( ) [static]`

Returns a pointer to the currently executing thread object.

#### Returns

**Pointer** (p. 2896) to the **Thread** (p. 3707) object representing the currently running **Thread** (p. 3707).

6.809.4.2 `static long long decaf::lang::Thread::getId ( ) [static]`

Obtains the **Thread** (p. 3707) Id of the current thread.

#### Returns

**Thread** (p. 3707) Id

6.809.4.3 `std::string decaf::lang::Thread::getName ( ) const`

Returns the Thread's assigned name.

#### Returns

the Name of the **Thread** (p. 3707).

6.809.4.4 `int decaf::lang::Thread::getPriority ( ) const`

Gets the currently set priority for this **Thread** (p. 3707).

#### Returns

an int value representing the Thread's current priority.

**6.809.4.5 Thread::State decaf::lang::Thread::getState ( ) const**

Returns the currently set State of this **Thread** (p. 3707).

**Returns**

the Thread's current state.

**6.809.4.6 const UncaughtExceptionHandler\* decaf::lang::Thread::getUncaughtExceptionHandler ( ) const**

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

**Returns**

a pointer to the set **UncaughtExceptionHandler** (p. 3841).

**6.809.4.7 bool decaf::lang::Thread::isAlive ( ) const**

Returns true if the **Thread** (p. 3707) is alive, meaning it has been started and has not yet died.

**Returns**

true if the thread is alive.

**6.809.4.8 virtual void decaf::lang::Thread::join ( long long *millisecs*, unsigned int *nanos* ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException ) [virtual]**

Forces the Current **Thread** (p. 3707) to wait until the thread exits.

**Parameters**

<i>millisecs</i>	the time in Milliseconds before the thread resumes
<i>nanos</i>	0-999999 extra nanoseconds to sleep.

**Exceptions**

<i>IllegalArgumentEx-ception</i>	if the nanoseconds parameter is out of range or the milliseconds paramter is negative.
<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.809.4.9 `virtual void decaf::lang::Thread::join ( ) throw ( decaf::lang::exceptions::InterruptedException )` [virtual]

Forces the Current **Thread** (p. 3707) to wait until the thread exits.

#### Exceptions

<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.
-----------------------------	--

6.809.4.10 `virtual void decaf::lang::Thread::join ( long long millisecs ) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::InterruptedException )` [virtual]

Forces the Current **Thread** (p. 3707) to wait until the thread exits.

#### Parameters

<i>millisecs</i>	the time in Milliseconds before the thread resumes
------------------	--

#### Exceptions

<i>IllegalArgumentException</i>	if the milliseconds parameter is negative.
<i>InterruptedException</i>	if any thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

6.809.4.11 `virtual void decaf::lang::Thread::run ( )` [virtual]

Default implementation of the run method - does nothing.

Implements **decaf::lang::Runnable** (p. 3265).

Reimplemented in **activemq::transport::mock::InternalCommandListener** (p. 2086), and **decaf::util::concurrent::PooledThread** (p. 2919).

6.809.4.12 `void decaf::lang::Thread::setName ( const std::string & name )`

Sets the name of the **Thread** (p. 3707) to the new Name given by the argument *name* name the new name of the **Thread** (p. 3707).

6.809.4.13 `void decaf::lang::Thread::setPriority ( int value ) throw ( decaf::lang::exceptions::IllegalArgumentException )`

Sets the current Thread's priority to the newly specified value.

The given value must be within the range **Thread::MIN\_PRIORITY** (p. 3716) and **Thread::MAX\_PRIORITY** (p. 3716).

#### Parameters

<i>value</i>	the new priority value to assign to this <b>Thread</b> (p. 3707).
--------------	---

#### Exceptions

<i>IllegalArgumentException</i>	if the value is out of range.
---------------------------------	-------------------------------

#### 6.809.4.14 void decaf::lang::Thread::setUncaughtExceptionHandler ( **UncaughtExceptionHandler** \* *handler* )

Set the handler invoked when this thread abruptly terminates due to an uncaught exception.

#### Parameters

<i>handler</i>	the <b>UncaughtExceptionHandler</b> to invoke when the <b>Thread</b> (p. 3707) terminates due to an uncaught exception.
----------------	---

#### 6.809.4.15 static void decaf::lang::Thread::sleep ( long long *millisecs*, unsigned int *nanos* ) throw ( **lang::exceptions::InterruptedException**, **lang::exceptions::IllegalArgumentException** ) [static]

Causes the currently executing thread to halt execution for the specified number of milliseconds plus any additionally specified nanoseconds given, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

#### Parameters

<i>millisecs</i>	time in milliseconds to halt execution.
<i>nanos</i>	0-999999 extra nanoseconds to sleep.

#### Exceptions

<i>IllegalArgumentException</i>	if the nanoseconds parameter is out of range or the milliseconds parameter is negative.
<i>InterruptedException</i>	if the <b>Thread</b> (p. 3707) was interrupted while sleeping.

6.809.4.16 `static void decaf::lang::Thread::sleep ( long long millisecs  
                   ) throw ( lang::exceptions::InterruptedException,  
                   lang::exceptions::IllegalArgumentException ) [static]`

Causes the currently executing thread to halt execution for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Note that this method is a static method that applies to the calling thread and not to the thread object.

#### Parameters

<i>millisecs</i>	time in milliseconds to halt execution.
------------------	---

#### Exceptions

<i>IllegalArgumentException</i>	if the milliseconds parameter is negative.
<i>InterruptedException</i>	if the <b>Thread</b> (p. 3707) was interrupted while sleeping.

6.809.4.17 `virtual void decaf::lang::Thread::start ( ) throw (  
                   decaf::lang::exceptions::IllegalThreadStateException,  
                   decaf::lang::exceptions::RuntimeException ) [virtual]`

Creates a system thread and starts it in a joinable mode.

Upon creation, the **run()** (p. 3713) method of either this object or the provided **Runnable** (p. 3264) object will be invoked in the context of this thread.

#### Exceptions

<i>IllegalThreadStateException</i>	if the thread has already been started.
<i>RuntimeException</i>	if the <b>Thread</b> (p. 3707) cannot be created for some reason.

6.809.4.18 `std::string decaf::lang::Thread::toString ( ) const`

Returns a string that describes the **Thread** (p. 3707).

#### Returns

string describing the **Thread** (p. 3707).

6.809.4.19 `static void decaf::lang::Thread::yield ( ) [static]`

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

### 6.809.5 Friends And Related Function Documentation

6.809.5.1 friend class `decaf::lang::Runtime` [`friend`]

6.809.5.2 friend class `decaf::util::concurrent::locks::LockSupport` [`friend`]

### 6.809.6 Field Documentation

6.809.6.1 `const int decaf::lang::Thread::MAX_PRIORITY = 10` [`static`]

The maximum priority that a thread can have.

6.809.6.2 `const int decaf::lang::Thread::MIN_PRIORITY = 1` [`static`]

The minimum priority that a thread can have.

6.809.6.3 `const int decaf::lang::Thread::NORM_PRIORITY = 5` [`static`]

The default priority that a thread is given at create time.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Thread.h`

## 6.810 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p. 3716)

```
#include <src/main/decaf/util/concurrent/ThreadFactory.h>
```

### Public Member Functions

- virtual `~ThreadFactory ()`
- virtual `decaf::lang::Thread * newThread (decaf::lang::Runnable *r)=0`

*Constructs a new Thread.*

### 6.810.1 Detailed Description

public interface **ThreadFactory** (p. 3716)

An object that creates new threads on demand. Using thread factories removes hard-wiring of calls to `new Thread`, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

```
class SimpleThreadFactory : public ThreadFactory (p. 3716) { public: Thread* newThread(
Runnable* r ) { return new Thread(r); } }
```

The `Executors.defaultThreadFactory()` method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

#### Since

1.0

### 6.810.2 Constructor & Destructor Documentation

6.810.2.1 `virtual decaf::util::concurrent::ThreadFactory::~~ThreadFactory ( ) [inline, virtual]`

### 6.810.3 Member Function Documentation

6.810.3.1 `virtual decaf::lang::Thread* decaf::util::concurrent::ThreadFactory::newThread ( decaf::lang::Runnable * r ) [pure virtual]`

Constructs a new Thread.

Implementations may also initialize priority, name, daemon status, ThreadGroup, etc. The pointer passed is still owned by the caller and is not deleted by the Thread object. The caller owns the returned Thread object and must delete it when finished.

#### Parameters

<i>r</i>	A pointer to a Runnable instance to be executed by new Thread instance returned.
----------	--

#### Returns

constructed thread, or NULL if the request to create a thread is rejected the caller owns the returned pointer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadFactory.h`

## 6.811 decaf::lang::ThreadGroup Class Reference

```
#include <src/main/decaf/lang/ThreadGroup.h>
```

### Public Member Functions

- `ThreadGroup ()`
- `virtual ~ThreadGroup ()`



### 6.811.1 Detailed Description

Since

1.0

### 6.811.2 Constructor & Destructor Documentation

6.811.2.1 decaf::lang::ThreadGroup::ThreadGroup ( )

6.811.2.2 virtual decaf::lang::ThreadGroup::~~ThreadGroup ( ) [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/ThreadGroup.h

## 6.812 decaf::util::concurrent::ThreadPool Class Reference

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

```
#include <src/main/decaf/util/concurrent/ThreadPool.h>
```

Inheritance diagram for decaf::util::concurrent::ThreadPool:

### Public Types

- typedef std::pair< lang::Runnable \*, TaskListener \* > Task

### Public Member Functions

- **ThreadPool** ()
- virtual **~ThreadPool** ()
- virtual void **queueTask** (Task task) throw ( lang::Exception )  
*Queue* (p. 3094) a task to be completed by one of the Pooled Threads.
- virtual **Task deQueueTask** () throw ( lang::Exception )  
*DeQueue* a task to be completed by one of the Pooled Threads.
- virtual std::size\_t **getPoolSize** () const  
*Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.*
- virtual std::size\_t **getBacklog** () const  
*Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.*
- virtual void **reserve** (std::size\_t size)

*Ensures that there is at least the specified number of Threads allocated to the pool.*

- virtual std::size\_t **getMaxThreads** () const

*Get the Max Number of Threads this Pool can contain.*

- virtual void **setMaxThreads** (std::size\_t maxThreads)

*Sets the Max number of threads this pool can contain.*

- virtual std::size\_t **getBlockSize** () const

*Gets the Max number of threads that can be allocated at a time when new threads are needed.*

- virtual void **setBlockSize** (std::size\_t blockSize)

*Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.*

- virtual std::size\_t **getFreeThreadCount** () const

*Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.*

- virtual void **onTaskStarted** (PooledThread \*thread)

*Called by a pooled thread when it is about to begin executing a new task.*

- virtual void **onTaskCompleted** (PooledThread \*thread)

*Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.*

- virtual void **onTaskException** (PooledThread \*thread, lang::Exception &ex)

*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2918) is now no longer running.*

## Static Public Member Functions

- static ThreadPool \* **getInstance** ()

*Return the one and only Thread Pool instance.*

## Static Public Attributes

- static const size\_t **DEFAULT\_MAX\_POOL\_SIZE** = 10
- static const size\_t **DEFAULT\_MAX\_BLOCK\_SIZE** = 3

### 6.812.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

The Thread Pool has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the `Runnable` interface and one of the worker threads will executing it in its thread context.

## 6.812.2 Member Typedef Documentation

6.812.2.1 `typedef std::pair<lang::Runnable*, TaskListener*>  
decaf::util::concurrent::ThreadPool::Task`

## 6.812.3 Constructor & Destructor Documentation

6.812.3.1 `decaf::util::concurrent::ThreadPool::ThreadPool ( )`

6.812.3.2 `virtual decaf::util::concurrent::ThreadPool::~~ThreadPool ( ) [virtual]`

## 6.812.4 Member Function Documentation

6.812.4.1 `virtual Task decaf::util::concurrent::ThreadPool::deQueueTask ( ) throw (  
lang::Exception) [virtual]`

DeQueue a task to be completed by one of the Pooled Threads.

A caller of this method will block until there is something in the tasks queue, therefore care must be taken when calling this function. Normally clients of **ThreadPool** (p. 3718) don't use this, only the **PooledThread** (p. 2918) objects owned by this **ThreadPool** (p. 3718).

### Returns

object that derives from Runnable

### Exceptions

<i>ActiveMQException</i>
--------------------------

6.812.4.2 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBacklog ( ) const  
[inline, virtual]`

Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

### Returns

number of outstanding tasks.

**6.812.4.3** `virtual std::size_t decaf::util::concurrent::ThreadPool::getBlockSize ( ) const`  
[inline, virtual]

Gets the Max number of threads that can be allocated at a time when new threads are needed.

#### Returns

max Thread Block Size

**6.812.4.4** `virtual std::size_t decaf::util::concurrent::ThreadPool::getFreeThreadCount ( ) const`  
[inline, virtual]

Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.

This value could change immediately after calling as Threads could finish right after and be available again. This is informational only.

#### Returns

total free threads

**6.812.4.5** `static ThreadPool* decaf::util::concurrent::ThreadPool::getInstance ( )`  
[static]

Return the one and only Thread Pool instance.

#### Returns

The Thread Pool Pointer

**6.812.4.6** `virtual std::size_t decaf::util::concurrent::ThreadPool::getMaxThreads ( ) const`  
[inline, virtual]

Get the Max Number of Threads this Pool can contain.

#### Returns

max size

**6.812.4.7** `virtual std::size_t decaf::util::concurrent::ThreadPool::getPoolSize ( ) const`  
[inline, virtual]

Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

## Returns

integer number of threads in existence.

6.812.4.8 `virtual void decaf::util::concurrent::ThreadPool::onTaskCompleted ( PooledThread * thread ) [virtual]`

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

## Parameters

<i>thread</i>	Pointer the the Pooled Thread that is making this call.
---------------	---

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2921).

6.812.4.9 `virtual void decaf::util::concurrent::ThreadPool::onTaskException ( PooledThread * thread, lang::Exception & ex ) [virtual]`

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2918) is now no longer running.

## Parameters

<i>thread</i>	Pointer to the Pooled Thread that is making this call
<i>ex</i>	The Exception that occurred.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2921).

6.812.4.10 `virtual void decaf::util::concurrent::ThreadPool::onTaskStarted ( PooledThread * thread ) [virtual]`

Called by a pooled thread when it is about to begin executing a new task.

This will decrement the available threads counter so that this object knows when there are no more free threads and must create new ones.

## Parameters

<i>thread</i>	Pointer to the Pooled Thread that is making this call
---------------	---

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2922).

6.812.4.11 `virtual void decaf::util::concurrent::ThreadPool::queueTask ( Task task ) throw ( lang::Exception ) [virtual]`

**Queue** (p. 3094) a task to be completed by one of the Pooled Threads.

tasks are serviced as soon as a **PooledThread** (p. 2918) is available to run it.

#### Parameters

<i>task</i>	object that derives from Runnable
-------------	-----------------------------------

#### Exceptions

<i>ActiveMQException</i>	
--------------------------	--

6.812.4.12 `virtual void decaf::util::concurrent::ThreadPool::reserve ( std::size_t size )`  
[virtual]

Ensures that there is at least the specified number of Threads allocated to the pool.

If the size is greater than the MAX number of threads in the pool, then only MAX threads are reservved. If the size is smaller than the number of threads currently in the pool, than nothing is done.

#### Parameters

<i>size</i>	the number of threads to reserve.
-------------	-----------------------------------

6.812.4.13 `virtual void decaf::util::concurrent::ThreadPool::setBlockSize ( std::size_t blockSize )`  
[virtual]

Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

#### Parameters

<i>blockSize</i>	Max Thread Block Size
------------------	-----------------------

6.812.4.14 `virtual void decaf::util::concurrent::ThreadPool::setMaxThreads ( std::size_t maxThreads )`  
[virtual]

Sets the Max number of threads this pool can contian.

if this value is smaller than the current size of the pool nothing is done.

#### Parameters

<i>maxThreads</i>	total number of threads that can be pooled
-------------------	--

### 6.812.5 Field Documentation

6.812.5.1 `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_BLOCK_SIZE = 3` [static]

6.812.5.2 `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_POOL_SIZE = 10` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadPool.h`

## 6.813 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

```
#include <src/main/decaf/lang/Throwable.h>
```

Inheritance diagram for decaf::lang::Throwable:

### Public Member Functions

- **Throwable** () throw ()
- virtual **~Throwable** () throw ()
- virtual std::string **getMessage** () const =0  
*Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.*
- virtual const std::exception \* **getCause** () const =0  
*Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*
- virtual void **initCause** (const std::exception \*cause)=0  
*Initializes the contained cause exception with the one given.*
- virtual void **setMark** (const char \*file, const int lineNumber)=0  
*Adds a file/line number to the stack trace.*
- virtual **Throwable** \* **clone** () const =0  
*Clones this exception.*
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const =0  
*Provides the stack trace for every point where this exception was caught, marked, and rethrown.*
- virtual void **printStackTrace** () const =0  
*Prints the stack trace to std::err.*
- virtual void **printStackTrace** (std::ostream &stream) const =0  
*Prints the stack trace to the given output stream.*
- virtual std::string **getStackTraceString** () const =0  
*Gets the stack trace as one contiguous string.*

### 6.813.1 Detailed Description

This class represents an error that has occurred.

All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1794) class in order to ensure that all Decaf Exceptions are interchangeable with the `std::exception` class.

**Throwable** (p. 3724) can wrap another **Throwable** (p. 3724) as the cause if the error being thrown. The user can inspect the cause by calling `getCause`, the pointer returned is the property of the **Throwable** (p. 3724) instance and will be deleted when it is deleted or goes out of scope.

#### Since

1.0

### 6.813.2 Constructor & Destructor Documentation

6.813.2.1 `decaf::lang::Throwable::Throwable ( ) throw () [inline]`

6.813.2.2 `virtual decaf::lang::Throwable::~~Throwable ( ) throw () [inline, virtual]`

### 6.813.3 Member Function Documentation

6.813.3.1 `virtual Throwable* decaf::lang::Throwable::clone ( ) const [pure virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

Copy of this **Exception** (p. 1794) object

Implemented in **activemq::exceptions::ActiveMQException** (p. 330), **activemq::exceptions::BrokerException** (p. 828), **decaf::internal::net::ssl::openssl::OpenSSLSocketException** (p. 2824), **decaf::io::EOFException** (p. 1791), **decaf::io::InterruptedIOException** (p. 2091), **decaf::io::IOException** (p. 2105), **decaf::io::UnsupportedEncodingException** (p. 3849), **decaf::io::UTFDataFormatException** (p. 3900), **decaf::lang::Exception** (p. 1797), **decaf::lang::exceptions::ClassCastException** (p. 1119), **decaf::lang::exceptions::IllegalArgumentException** (p. 1955), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1957), **decaf::lang::exceptions::IllegalStateException** (p. 1961), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1964), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1970), **decaf::lang::exceptions::InterruptedException** (p. 2089), **decaf::lang::exceptions::InvalidStateException** (p. 2102), **decaf::lang::exceptions::NoSuchElementException** (p. 2781), **decaf::lang::exceptions::NullPointerException** (p. 2786), **decaf::lang::exceptions::NumberFormatException** (p. 2791), **decaf::lang::exceptions::RuntimeException** (p. 3269), **decaf::lang::exceptions::UnsupportedOperationException** (p. 3852), **decaf::net::BindException** (p. 800), **decaf::net::ConnectException** (p. 1232), **decaf::net::HttpRetryException**



(p. 1950), **decaf::net::MalformedURLException** (p. 2418), **decaf::net::NoRouteToHostException** (p. 2775), **decaf::net::PortUnreachableException** (p. 2924), **decaf::net::ProtocolException** (p. 3085), **decaf::net::SocketException** (p. 3467), **decaf::net::SocketTimeoutException** (p. 3489), **decaf::net::UnknownHostException** (p. 3844), **decaf::net::UnknownServiceException** (p. 3846), **decaf::net::URISyntaxException** (p. 3883), **decaf::nio::BufferOverflowException** (p. 916), **decaf::nio::BufferUnderflowException** (p. 918), **decaf::nio::InvalidMarkException** (p. 2099), **decaf::nio::ReadOnlyBufferException** (p. 3117), **decaf::security::cert::CertificateEncodingException** (p. 1061), **decaf::security::cert::CertificateException** (p. 1062), **decaf::security::cert::CertificateExpiredException** (p. 1064), **decaf::security::cert::CertificateNotYetValidException** (p. 1066), **decaf::security::cert::CertificateParsingException** (p. 1068), **decaf::security::GeneralSecurityException** (p. 1936), **decaf::security::InvalidKeyException** (p. 2096), **decaf::security::KeyException** (p. 2257), **decaf::security::KeyManagementException** (p. 2260), **decaf::security::NoSuchAlgorithmException** (p. 2778), **decaf::security::NoSuchProviderException** (p. 2783), **decaf::security::SignatureException** (p. 3442), **decaf::util::concurrent::BrokenBarrierException** (p. 823), **decaf::util::concurrent::CancellationException** (p. 1055), **decaf::util::concurrent::ExecutionException** (p. 1831), **decaf::util::concurrent::RejectedExecutionException** (p. 3136), **decaf::util::concurrent::TimeoutException** (p. 3730), **decaf::util::zip::DataFormatException** (p. 1522), and **decaf::util::zip::ZipException** (p. 3993).

**6.813.3.2** `virtual const std::exception* decaf::lang::Throwable::getCause ( ) const [pure virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

### Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1798).

**6.813.3.3** `virtual std::string decaf::lang::Throwable::getMessage ( ) const [pure virtual]`

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value `cause.getMessage` is then returned.

### Returns

string errors message

Implemented in **decaf::lang::Exception** (p. 1798).

6.813.3.4 `virtual std::vector< std::pair< std::string, int> >  
 decaf::lang::Throwable::getStackTrace ( ) const [pure virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

#### Returns

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1798).

6.813.3.5 `virtual std::string decaf::lang::Throwable::getStackTraceString ( ) const [pure  
 virtual]`

Gets the stack trace as one contiguous string.

#### Returns

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1798).

6.813.3.6 `virtual void decaf::lang::Throwable::initCause ( const std::exception * cause )  
 [pure virtual]`

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

#### Parameters

<i>cause</i>	The exception that was the cause of this one.
--------------	---

Implemented in **decaf::lang::Exception** (p. 1799).

6.813.3.7 `virtual void decaf::lang::Throwable::printStackTrace ( ) const [pure  
 virtual]`

Prints the stack trace to `std::err`.

Implemented in **decaf::lang::Exception** (p. 1799).

6.813.3.8 `virtual void decaf::lang::Throwable::printStackTrace ( std::ostream & stream ) const  
 [pure virtual]`

Prints the stack trace to the given output stream.

**Parameters**

<i>stream</i>	the target output stream.
---------------	---------------------------

Implemented in **decaf::lang::Exception** (p. 1799).

6.813.3.9 virtual void decaf::lang::Throwable::setMark ( const char \* *file*, const int *lineNumber* ) [pure virtual]

Adds a file/line number to the stack trace.

**Parameters**

<i>file</i>	The name of the file calling this method (use __FILE__).
<i>lineNumber</i>	The line number in the calling file (use __LINE__).

Implemented in **decaf::lang::Exception** (p. 1799).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Throwable.h

**6.814 decaf::util::concurrent::TimeoutException Class Reference**

```
#include <src/main/decaf/util/concurrent/TimeoutException.h>
```

Inheritance diagram for decaf::util::concurrent::TimeoutException:

**Public Member Functions**

- **TimeoutException** () throw ()  
*Default Constructor.*
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **TimeoutException** (const **TimeoutException** &ex) throw ()  
*Copy Constructor.*
- **TimeoutException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **TimeoutException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **TimeoutException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **TimeoutException** \* **clone** () const  
*Clones this exception.*
- virtual ~**TimeoutException** () throw ()

### 6.814.1 Constructor & Destructor Documentation

6.814.1.1 **decaf::util::concurrent::TimeoutException::TimeoutException** ( ) throw ()  
[inline]

Default Constructor.

6.814.1.2 **decaf::util::concurrent::TimeoutException::TimeoutException** ( const **decaf::lang::Exception** & *ex* ) throw () [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.814.1.3 **decaf::util::concurrent::TimeoutException::TimeoutException** ( const **TimeoutException** & *ex* ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	The exception to copy from.
-----------	-----------------------------

6.814.1.4 **decaf::util::concurrent::TimeoutException::TimeoutException** ( const std::exception \* *cause* ) throw () [inline]

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.814.1.5 **decaf::util::concurrent::TimeoutException::TimeoutException** ( const char \* *file*, const int *lineNumber*, const char \* *msg*, ... ) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the mes-

sage

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The string message to report
...	list of primitives that are formatted into the message

**6.814.1.6** `decaf::util::concurrent::TimeoutException::TimeoutException ( const char * file,  
const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The string message to report
...	list of primitives that are formatted into the message

**6.814.1.7** `virtual decaf::util::concurrent::TimeoutException::~~TimeoutException ( ) throw ()`  
[inline, virtual]

## 6.814.2 Member Function Documentation

**6.814.2.1** `virtual TimeoutException* decaf::util::concurrent::TimeoutException::clone ( )`  
`const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new **TimeoutException** (p. 3728) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeoutException.h`

## 6.815 decaf::util::Timer Class Reference

A facility for threads to schedule tasks for future execution in a background thread.

```
#include <src/main/decaf/util/Timer.h>
```

### Public Member Functions

- **Timer** ()
- virtual **~Timer** ()
- void **cancel** ()  
*Terminates this timer, discarding any currently scheduled tasks.*
- std::size\_t **purge** ()  
*Removes all canceled tasks from this timer's task queue.*
- void **schedule** (**TimerTask** \*task, long long delay) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution after the specified delay.*
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution after the specified delay.*
- void **schedule** (**TimerTask** \*task, const **Date** &time) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution at the specified time.*
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &time) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for execution at the specified time.*
- void **schedule** (**TimerTask** \*task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.*
- void **schedule** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.*
- void **schedule** (**TimerTask** \*task, const **Date** &firstTime, long long period) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )  
*Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.*

- void **schedule** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &firstTime, long long period) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalArgumentException**, **decaf::lang::exceptions::IllegalStateException** )  
*Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.*
- void **scheduleAtFixedRate** (**TimerTask** \*task, long long delay, long long period) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalArgumentException**, **decaf::lang::exceptions::IllegalStateException** )  
*Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.*
- void **scheduleAtFixedRate** (const **decaf::lang::Pointer**< **TimerTask** > &task, long long delay, long long period) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalArgumentException**, **decaf::lang::exceptions::IllegalStateException** )  
*Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.*
- void **scheduleAtFixedRate** (**TimerTask** \*task, const **Date** &firstTime, long long period) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalArgumentException**, **decaf::lang::exceptions::IllegalStateException** )  
*Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.*
- void **scheduleAtFixedRate** (const **decaf::lang::Pointer**< **TimerTask** > &task, const **Date** &firstTime, long long period) throw ( **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalArgumentException**, **decaf::lang::exceptions::IllegalStateException** )  
*Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.*

### 6.815.1 Detailed Description

A facility for threads to schedule tasks for future execution in a background thread.

Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each **Timer** (p. 3730) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 3730) tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

This class is thread-safe: multiple threads can share a single **Timer** (p. 3730) object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the wait(long) method.

**Since**

1.0

**6.815.2 Constructor & Destructor Documentation****6.815.2.1** `decaf::util::Timer::Timer ( )`**6.815.2.2** `virtual decaf::util::Timer::~~Timer ( )` [virtual]**6.815.3 Member Function Documentation****6.815.3.1** `void decaf::util::Timer::cancel ( )`

Terminates this timer, discarding any currently scheduled tasks.

Does not interfere with a currently executing task (if it exists). Once a timer has been terminated, its execution thread terminates gracefully, and no more tasks may be scheduled on it.

Note that calling this method from within the run method of a timer task that was invoked by this timer absolutely guarantees that the ongoing task execution is the last task execution that will ever be performed by this timer.

This method may be called repeatedly; the second and subsequent calls have no effect.

**6.815.3.2** `std::size_t decaf::util::Timer::purge ( )`

Removes all canceled tasks from this timer's task queue.

Calling this method has no effect on the behavior of the timer, but eliminates the canceled tasks from the queue causing the **Timer** (p.3730) to destroy the **TimerTask** (p.3743) pointer it was originally given, the caller should ensure that they no longer have any references to TimerTasks that were previously scheduled.

Most programs will have no need to call this method. It is designed for use by the rare application that cancels a large number of tasks. Calling this method trades time for space: the runtime of the method may be proportional to  $n + c \log n$ , where  $n$  is the number of tasks in the queue and  $c$  is the number of canceled tasks.

This method can be called on a **Timer** (p.3730) object that has no scheduled tasks without error.

**Returns**

the number of tasks removed from the queue.



```
6.815.3.3 void decaf::util::Timer::schedule ( const decaf::lang::Pointer<
    TimerTask > & task, long long delay ) throw (
    decaf::lang::exceptions::NullPointerException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution after the specified delay.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, or timer was cancelled.

```
6.815.3.4 void decaf::util::Timer::schedule ( const decaf::lang::Pointer<
    TimerTask > & task, long long delay, long long period )
    throw ( decaf::lang::exceptions::NullPointerException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

**6.815.3.5** void decaf::util::Timer::schedule ( **TimerTask** \* *task*, const Date & *firstTime*, long long *period* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3743) pointer is considered to be owned by the **Timer** (p. 3730) class once it has been scheduled, the **Timer** (p. 3730) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3730) itself is cancelled. A **TimerTask** (p. 3743) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3730) and the caller should ensure that the **TimerTask** (p. 3743) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3743) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

**Parameters**

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.

<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.
------------------------------	--

6.815.3.6 void decaf::util::Timer::schedule ( **TimerTask** \* *task*, const Date & *time* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

The **TimerTask** (p. 3743) pointer is considered to be owned by the **Timer** (p. 3730) class once it has been scheduled, the **Timer** (p. 3730) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3730) itself is cancelled. A **TimerTask** (p. 3743) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3730) and the caller should ensure that the **TimerTask** (p. 3743) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3743) instance are planned.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>time</i>	- time at which task is to be executed.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.815.3.7 void decaf::util::Timer::schedule ( const decaf::lang::Pointer< **TimerTask** > & *task*, const Date & *firstTime*, long long *period* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other

background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if <code>time.getTime()</code> is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.8 void decaf::util::Timer::schedule ( TimerTask * task, long long
      delay ) throw ( decaf::lang::exceptions::NullPointerException,
      decaf::lang::exceptions::IllegalArgumentException,
      decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution after the specified delay.

The **TimerTask** (p. 3743) pointer is considered to be owned by the **Timer** (p. 3730) class once it has been scheduled, the **Timer** (p. 3730) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3730) itself is cancelled. A **TimerTask** (p. 3743) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3730) and the caller should ensure that the **TimerTask** (p. 3743) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3743) instance are planned.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if <code>delay</code> is negative, or <code>delay + System.currentTimeMillis()</code> is negative.

<i>IllegalStateException</i>	- if task was already scheduled or cancelled, or timer was cancelled.
------------------------------	---

6.815.3.9 void decaf::util::Timer::schedule ( **TimerTask** \* *task*, long long *delay*, long long *period* ) throw ( decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException )

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3743) pointer is considered to be owned by the **Timer** (p. 3730) class once it has been scheduled, the **Timer** (p. 3730) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3730) itself is cancelled. A **TimerTask** (p. 3743) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3730) and the caller should ensure that the **TimerTask** (p. 3743) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3743) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + <code>System.currentTimeMillis()</code> is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.10 void decaf::util::Timer::schedule ( const decaf::lang::Pointer<
TimerTask > & task, const Date & time ) throw (
decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>time</i>	- time at which task is to be executed.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.11 void decaf::util::Timer::scheduleAtFixedRate ( TimerTask * task, long long delay,
long long period ) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3743) pointer is considered to be owned by the **Timer** (p. 3730) class once it has been scheduled, the **Timer** (p. 3730) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3730) itself is cancelled. A **TimerTask** (p. 3743) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3730) and the caller should ensure that the **TimerTask** (p. 3743) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3743) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance

every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.12 void decaf::util::Timer::scheduleAtFixedRate ( const decaf::lang::Pointer<
TimerTask > & task, long long delay, long long period )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

### Parameters

<i>task</i>	- task to be scheduled.
<i>delay</i>	- delay in milliseconds before task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if delay is negative, or delay + System.currentTimeMillis() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.815.3.13 void decaf::util::Timer::scheduleAtFixedRate ( const decaf::lang::Pointer<
TimerTask > & task, const Date & firstTime, long long period
) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

**Parameters**

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

**Exceptions**

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if time.getTime() is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.



```
6.815.3.14 void decaf::util::Timer::scheduleAtFixedRate ( TimerTask
* task, const Date & firstTime, long long period ) throw
( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3743) pointer is considered to be owned by the **Timer** (p. 3730) class once it has been scheduled, the **Timer** (p. 3730) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 3730) itself is cancelled. A **TimerTask** (p. 3743) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 3730) and the caller should ensure that the **TimerTask** (p. 3743) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3743) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

#### Parameters

<i>task</i>	- task to be scheduled.
<i>firstTime</i>	- First time at which task is to be executed.
<i>period</i>	- time in milliseconds between successive task executions.

#### Exceptions

<i>NullPointerException</i>	- if the <b>TimerTask</b> (p. 3743) value is Null.
<i>IllegalArgumentException</i>	- if <code>time.getTime()</code> is negative.
<i>IllegalStateException</i>	- if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Timer.h`

## 6.816 decaf::util::TimerTask Class Reference

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3730).

```
#include <src/main/decaf/util/TimerTask.h>
```

Inheritance diagram for decaf::util::TimerTask:

### Public Member Functions

- **TimerTask** ()
- virtual **~TimerTask** ()
- bool **cancel** ()  
*Cancels this timer task.*
- long long **scheduledExecutionTime** () const  
*Returns the scheduled execution time of the most recent actual execution of this task.*

### Protected Member Functions

- bool **isScheduled** () const
- void **setScheduledTime** (long long time)
- long long **getWhen** () const

### Friends

- class **Timer**
- class **TimerImpl**
- class **decaf::internal::util::TimerTaskHeap**

#### 6.816.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3730).

#### Since

1.0

#### 6.816.2 Constructor & Destructor Documentation

##### 6.816.2.1 decaf::util::TimerTask::TimerTask ( )

6.816.2.2 `virtual decaf::util::TimerTask::~~TimerTask ( ) [inline, virtual]`

### 6.816.3 Member Function Documentation

6.816.3.1 `bool decaf::util::TimerTask::cancel ( )`

Cancels this timer task.

If the task has been scheduled for one-time execution and has not yet run, or has not yet been scheduled, it will never run. If the task has been scheduled for repeated execution, it will never run again. (If the task is running when this call occurs, the task will run to completion, but will never run again.)

Note that calling this method from within the run method of a repeating timer task absolutely guarantees that the timer task will not run again.

This method may be called repeatedly; the second and subsequent calls have no effect.

#### Returns

true if this task is scheduled for one-time execution and has not yet run, or this task is scheduled for repeated execution. Returns false if the task was scheduled for one-time execution and has already run, or if the task was never scheduled, or if the task was already canceled. (Loosely speaking, this method returns true if it prevents one or more scheduled executions from taking place.)

6.816.3.2 `long long decaf::util::TimerTask::getWhen ( ) const [protected]`

6.816.3.3 `bool decaf::util::TimerTask::isScheduled ( ) const [protected]`

6.816.3.4 `long long decaf::util::TimerTask::scheduledExecutionTime ( ) const`

Returns the scheduled execution time of the most recent actual execution of this task.

(If this method is invoked while task execution is in progress, the return value is the scheduled execution time of the ongoing task execution.)

This method is typically invoked from within a task's run method, to determine whether the current execution of the task is sufficiently timely to warrant performing the scheduled activity:

```
void run() (p. 3265) { if( System::currentTimeMillis() - scheduledExecutionTime() (p. 3744)
>= MAX_TARDINESS) return; // Too late; skip this execution. // Perform the task }
```

This method is typically not used in conjunction with fixed-delay execution repeating tasks, as their scheduled execution times are allowed to drift over time, and so are not terribly significant.

#### Returns

the time at which the most recent execution of this task was scheduled to occur, in the format returned by **Date.getTime()** (p. 1636). The return value is undefined if the task has yet to commence its first execution.

6.816.3.5 `void decaf::util::TimerTask::setScheduledTime ( long long time )` `[protected]`

## 6.816.4 Friends And Related Function Documentation

6.816.4.1 `friend class decaf::internal::util::TimerTaskHeap` `[friend]`

6.816.4.2 `friend class Timer` `[friend]`

6.816.4.3 `friend class TimerImpl` `[friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/TimerTask.h`

## 6.817 decaf::internal::util::TimerTaskHeap Class Reference

A Binary Heap implemented specifically for the Timer class in Decaf Util.

```
#include <src/main/decaf/internal/util/TimerTaskHeap.h>
```

### Public Member Functions

- **TimerTaskHeap** ()
- virtual **~TimerTaskHeap** ()
- **Pointer< TimerTask > peek** ()  
*Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.*
- bool **isEmpty** () const
- std::size\_t **size** () const
- void **insert** (const **Pointer< TimerTask >** &task)  
*Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.*
- void **remove** (std::size\_t pos)  
*Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.*
- void **reset** ()  
*Clear all contents from the heap.*
- void **adjustMinimum** ()  
*Resorts the heap starting at the top.*
- std::size\_t **deleteIfCancelled** ()  
*Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a a cancellation of a large number of tasks the user can perform this purge.*
- std::size\_t **find** (const **Pointer< TimerTask >** &task) const  
*Searches the heap for the specified TimerTask element and returns its position in the heap.*

### 6.817.1 Detailed Description

A Binary Heap implemented specifically for the Timer class in Decaf Util.

#### Since

1.0

### 6.817.2 Constructor & Destructor Documentation

6.817.2.1 `decaf::internal::util::TimerTaskHeap::TimerTaskHeap ( )`

6.817.2.2 `virtual decaf::internal::util::TimerTaskHeap::~~TimerTaskHeap ( )` [virtual]

### 6.817.3 Member Function Documentation

6.817.3.1 `void decaf::internal::util::TimerTaskHeap::adjustMinimum ( )`

Resorts the heap starting at the top.

6.817.3.2 `std::size_t decaf::internal::util::TimerTaskHeap::deletelfCancelled ( )`

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.

#### Returns

the number of task that were removed from the heap because they were cancelled.

6.817.3.3 `std::size_t decaf::internal::util::TimerTaskHeap::find ( const Pointer< TimerTask > & task ) const`

Searches the heap for the specified TimerTask element and returns its position in the heap.

Returns the unsigned equivalent of -1 if the element is not found.

#### Returns

the position in the Heap where the Task is stored, or npos.

6.817.3.4 `void decaf::internal::util::TimerTaskHeap::insert ( const Pointer< TimerTask > & task )`

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

**Parameters**

<i>task</i>	The TimerTask to insert into the heap.
-------------	--

6.817.3.5 **bool** decaf::internal::util::TimerTaskHeap::isEmpty ( ) const

**Returns**

true if the heap is empty.

6.817.3.6 **Pointer<TimerTask>** decaf::internal::util::TimerTaskHeap::peek ( )

Peaks at the Head of the Heap, returns the task with the nearest scheduled run time.

**Returns**

The TimerTask that is scheduled to be executed next if the Heap is empty a Null Pointer value is returned.

6.817.3.7 **void** decaf::internal::util::TimerTaskHeap::remove ( std::size\_t *pos* )

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

**Parameters**

<i>pos</i>	The position at which to remove the TimerTask and begin a resort of the heap.
------------	---

6.817.3.8 **void** decaf::internal::util::TimerTaskHeap::reset ( )

Clear all contents from the heap.

6.817.3.9 **std::size\_t** decaf::internal::util::TimerTaskHeap::size ( ) const

**Returns**

the size of the heap.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/TimerTaskHeap.h

## 6.818 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 3748) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

```
#include <src/main/decaf/util/concurrent/TimeUnit.h>
```

Inheritance diagram for decaf::util::concurrent::TimeUnit:

### Public Member Functions

- virtual **~TimeUnit** ()
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const  
  
*Convert the given time duration in the given unit to this unit.*
- long long **toNanos** (long long duration) const  
*Equivalent to NANoseconds.convert(duration, this).*
- long long **toMicros** (long long duration) const  
*Equivalent to MICROSECONDS.convert(duration, this).*
- long long **toMillis** (long long duration) const  
*Equivalent to MILLISECONDS.convert(duration, this).*
- long long **toSeconds** (long long duration) const  
*Equivalent to SECONDS.convert(duration, this).*
- long long **toMinutes** (long long duration) const  
*Equivalent to MINUTES.convert(duration, this).*
- long long **toHours** (long long duration) const  
*Equivalent to HOURS.convert(duration, this).*
- long long **toDays** (long long duration) const  
*Equivalent to DAYS.convert(duration, this).*
- void **timedWait** (**Synchronizable** \*obj, long long timeout) const throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )  
*Perform a timed Object.wait using this time unit.*
- void **timedJoin** (decaf::lang::Thread \*thread, long long timeout) throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )  
*Perform a timed Thread.join using this time unit.*
- void **sleep** (long long timeout) const throw ( decaf::lang::exceptions::InterruptedException )  
*Perform a Thread.sleep using this unit.*
- virtual std::string **toString** () const  
*Converts the TimeUnit (p. 3748) type to the Name of the TimeUnit (p. 3748).*
- virtual int **compareTo** (const **TimeUnit** &value) const

*Compares this object with the specified object for order.*

- virtual bool **equals** (const **TimeUnit** &value) const
- virtual bool **operator==** (const **TimeUnit** &value) const

*Compares equality between this object and the one passed.*

- virtual bool **operator<** (const **TimeUnit** &value) const

*Compares this object to another and returns true if this object is considered to be less than the one passed.*

### Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name) throw ( decaf::lang::exceptions::IllegalArgumentE )

*Returns the **TimeUnit** (p. 3748) constant of this type with the specified name.*

### Static Public Attributes

- static const **TimeUnit** **NANOSECONDS**

*The Actual **TimeUnit** (p. 3748) enumerations.*

- static const **TimeUnit** **MICROSECONDS**
- static const **TimeUnit** **MILLISECONDS**
- static const **TimeUnit** **SECONDS**
- static const **TimeUnit** **MINUTES**
- static const **TimeUnit** **HOURS**
- static const **TimeUnit** **DAYS**
- static const **TimeUnit** \*const **values** []

*The An Array of **TimeUnit** (p. 3748) Instances.*

### Protected Member Functions

- **TimeUnit** (int index, const std::string &name)

*Hidden Constructor, this class can not be instantiated directly.*

### 6.818.1 Detailed Description

A **TimeUnit** (p. 3748) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

A **TimeUnit** (p. 3748) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.



A **TimeUnit** (p. 3748) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the lock is not available:

```
Lock (p. 2334) lock = ...; if ( lock.tryLock( 50, TimeUnit::MILLISECONDS (p. 3757) ) ) ...
```

while this code will timeout in 50 seconds:

```
Lock (p. 2334) lock = ...; if ( lock.tryLock( 50, TimeUnit::SECONDS (p. 3757) ) ) ...
```

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 3748).

## 6.818.2 Constructor & Destructor Documentation

**6.818.2.1** `decaf::util::concurrent::TimeUnit::TimeUnit ( int index, const std::string & name )`  
[protected]

Hidden Constructor, this class can not be instantiated directly.

### Parameters

<i>index</i>	- Index into the Time Unit set.
<i>name</i>	- Name of the unit type being represented.

**6.818.2.2** `virtual decaf::util::concurrent::TimeUnit::~~TimeUnit ( )` [inline, virtual]

## 6.818.3 Member Function Documentation

**6.818.3.1** `virtual int decaf::util::concurrent::TimeUnit::compareTo ( const TimeUnit & value )`  
`const` [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all *x* and *y*. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementor must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all *z*.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`.

Generally speaking, any class that implements the Comparable interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

#### Parameters

<i>value</i>	- the Object to be compared.
--------------	------------------------------

#### Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

**6.818.3.2** `long long decaf::util::concurrent::TimeUnit::convert ( long long sourceDuration,  
const TimeUnit & sourceUnit ) const`

Convert the given time duration in the given unit to this unit.

Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to Long.MIN\_VALUE if negative or Long.MAX\_VALUE if positive.

For example, to convert 10 minutes to milliseconds, use: `TimeUnit.MILLISECONDS.convert(10L, TimeUnit.MINUTES (p. 3757))`

#### Parameters

<i>sourceDuration</i>	- Duration value to convert.
<i>sourceUnit</i>	- Unit type of the source duration.

#### Returns

the converted duration in this unit, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.

**6.818.3.3** `virtual bool decaf::util::concurrent::TimeUnit::equals ( const TimeUnit & value )  
const [virtual]`

#### Returns

true if this value is considered equal to the passed value.

**6.818.3.4** `virtual bool decaf::util::concurrent::TimeUnit::operator< ( const TimeUnit & value )  
const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

```
6.818.3.5 virtual bool decaf::util::concurrent::TimeUnit::operator==( const TimeUnit & value )  
const [virtual]
```

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

```
6.818.3.6 void decaf::util::concurrent::TimeUnit::sleep ( long long timeout ) const throw (  
decaf::lang::exceptions::InterruptedException )
```

Perform a `Thread.sleep` using this unit.

This is a convenience method that converts time arguments into the form required by the `Thread.sleep` method.

#### Parameters

<i>timeout</i>	the minimum time to sleep
----------------	---------------------------

#### See also

`Thread::sleep`

```
6.818.3.7 void decaf::util::concurrent::TimeUnit::timedJoin ( decaf::lang::Thread * thread,  
long long timeout ) throw ( decaf::lang::exceptions::InterruptedException,  
decaf::lang::exceptions::NullPointerException )
```

Perform a timed `Thread.join` using this time unit.

This is a convenience method that converts time arguments into the form required by the `Thread.join` method.

#### Parameters

<i>thread</i>	the thread to wait for
<i>timeout</i>	the maximum time to wait

**Exceptions**

<i>InterruptedException</i>	if interrupted while waiting.
<i>NullPointerException</i>	if the thread object is null.

**See also**

Thread::join( long long, long long )

6.818.3.8 void decaf::util::concurrent::TimeUnit::timedWait ( Synchronizable \* *obj*, long long *timeout* ) const throw ( decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException )

Perform a timed `Object.wait` using this time unit.

This is a convenience method that converts timeout arguments into the form required by the `Object.wait` method.

For example, you could implement a blocking `poll` method (see **BlockingQueue.poll** (p. 809)) using:

```
Object poll( long long timeout, const TimeUnit& unit )
    throw( InterruptedException ) {

    while( empty ) {
        unit.timedWait( this, timeout );
        ...
    }
}
```

**Parameters**

<i>obj</i>	the object to wait on
<i>timeout</i>	the maximum time to wait.

**Exceptions**

<i>InterruptedException</i>	if interrupted while waiting.
<i>NullPointerException</i>	if the <b>Synchronizable</b> (p. 3644) object is null.

**See also**

Synchronizable::wait( long long, long long )

**6.818.3.9** `long long decaf::util::concurrent::TimeUnit::toDays ( long long duration ) const`  
[inline]

Equivalent to `DAYS.convert(duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration.

#### See also

**convert** (p. 3751)

**6.818.3.10** `long long decaf::util::concurrent::TimeUnit::toHours ( long long duration ) const`  
[inline]

Equivalent to `HOURS.convert(duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration.

#### See also

**convert** (p. 3751)

**6.818.3.11** `long long decaf::util::concurrent::TimeUnit::toMicros ( long long duration ) const`  
[inline]

Equivalent to `MICROSECONDS.convert(duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

#### See also

**convert** (p. 3751)

6.818.3.12 `long long decaf::util::concurrent::TimeUnit::toMillis ( long long duration ) const`  
`[inline]`

Equivalent to `MILLISECONDS.convert(duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

#### See also

**convert** (p. 3751)

6.818.3.13 `long long decaf::util::concurrent::TimeUnit::toMinutes ( long long duration ) const`  
`[inline]`

Equivalent to `MINUTES.convert(duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration.

#### See also

**convert** (p. 3751)

6.818.3.14 `long long decaf::util::concurrent::TimeUnit::toNanos ( long long duration ) const`  
`[inline]`

Equivalent to `NANOSECONDS.convert(duration, this)`.

#### Parameters

<i>duration</i>	the duration
-----------------	--------------

#### Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

**See also**

**convert** (p. 3751)

6.818.3.15 `long long decaf::util::concurrent::TimeUnit::toSeconds ( long long duration ) const`  
[inline]

Equivalent to `SECONDS.convert (duration, this)`.

**Parameters**

<i>duration</i>	the duration
-----------------	--------------

**Returns**

the converted duration.

**See also**

**convert** (p. 3751)

6.818.3.16 `virtual std::string decaf::util::concurrent::TimeUnit::toString ( ) const`  
[virtual]

Converts the **TimeUnit** (p. 3748) type to the Name of the **TimeUnit** (p. 3748).

**Returns**

String name of the **TimeUnit** (p. 3748)

6.818.3.17 `static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf ( const std::string  
& name ) throw ( decaf::lang::exceptions::IllegalArgumentException )`  
[static]

Returns the **TimeUnit** (p. 3748) constant of this type with the specified name.

The string must match exactly an identifier used to declare an **TimeUnit** (p. 3748) constant in this type. (Extraneous whitespace characters are not permitted.)

**Parameters**

<i>name</i>	The Name of the <b>TimeUnit</b> (p. 3748) constant to be returned.
-------------	--

**Returns**

A constant reference to the **TimeUnit** (p. 3748) Constant with the given name.

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if this enum type has no constant with the specified name
---------------------------------------	---

#### 6.818.4 Field Documentation

6.818.4.1 `const TimeUnit decaf::util::concurrent::TimeUnit::DAYS` [static]

6.818.4.2 `const TimeUnit decaf::util::concurrent::TimeUnit::HOURS` [static]

6.818.4.3 `const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS`  
[static]

6.818.4.4 `const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS`  
[static]

6.818.4.5 `const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES`  
[static]

6.818.4.6 `const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS`  
[static]

The Actual **TimeUnit** (p. 3748) enumerations.

6.818.4.7 `const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS`  
[static]

6.818.4.8 `const TimeUnit* const decaf::util::concurrent::TimeUnit::values[]`  
[static]

The An Array of **TimeUnit** (p. 3748) Instances.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeUnit.h`

#### 6.819 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

```
#include <src/main/cms/Topic.h>
```

Inheritance diagram for cms::Topic:



## Public Member Functions

- virtual `~Topic()`
- virtual `std::string getTopicName() const` `=0` throw ( `CMSEException` )

*Gets the name of this topic.*

### 6.819.1 Detailed Description

An interface encapsulating a provider-specific topic name.

A **Topic** (p. 3757) is a Publish / Subscribe type **Destination** (p. 1688). All Messages sent to a **Topic** (p. 3757) are broadcast to all Subscribers of that **Topic** (p. 3757) unless the Subscriber defines a **Message** (p. 2493) selector that filters out that **Message** (p. 2493).

#### Since

1.0

### 6.819.2 Constructor & Destructor Documentation

6.819.2.1 `virtual cms::Topic::~Topic( )` `[inline, virtual]`

### 6.819.3 Member Function Documentation

6.819.3.1 `virtual std::string cms::Topic::getTopicName( )` `const` throw ( `CMSEException` )  
`[pure virtual]`

Gets the name of this topic.

#### Returns

The topic name.

#### Exceptions

<b><i>CMSEException</i></b> (p. 1130)	- If an internal error occurs.
--	--------------------------------

Implemented in `activemq::commands::ActiveMQTopic` (p. 663).

The documentation for this class was generated from the following file:

- `src/main/cms/Topic.h`

## 6.820 activemq::state::Tracked Class Reference

```
#include <src/main/activemq/state/Tracked.h>
```

Inheritance diagram for `activemq::state::Tracked`:

### Public Member Functions

- **Tracked** ()
- **Tracked** (const **Pointer**< **decaf::lang::Runnable** > &runnable)
- virtual **~Tracked** ()
- void **onResponse** ()
- bool **isWaitingForResponse** () const

### 6.820.1 Constructor & Destructor Documentation

6.820.1.1 `activemq::state::Tracked::Tracked ( )` `[inline]`

6.820.1.2 `activemq::state::Tracked::Tracked ( const Pointer< decaf::lang::Runnable > &  
runnable )`

6.820.1.3 `virtual activemq::state::Tracked::~~Tracked ( )` `[inline, virtual]`

### 6.820.2 Member Function Documentation

6.820.2.1 `bool activemq::state::Tracked::isWaitingForResponse ( )` const `[inline]`

6.820.2.2 `void activemq::state::Tracked::onResponse ( )`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/Tracked.h`

## 6.821 `activemq::commands::TransactionId` Class Reference

```
#include <src/main/activemq/commands/TransactionId.h>
```

Inheritance diagram for `activemq::commands::TransactionId`:

### Public Types

- typedef **decaf::lang::PointerComparator**< **TransactionId** > **COMPARATOR**

## Public Member Functions

- **TransactionId** ()
- **TransactionId** (const **TransactionId** &other)
- virtual ~**TransactionId** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **TransactionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual int **compareTo** (const **TransactionId** &value) const
- virtual bool **equals** (const **TransactionId** &value) const
- virtual bool **operator==** (const **TransactionId** &value) const
- virtual bool **operator<** (const **TransactionId** &value) const
- **TransactionId** & **operator=** (const **TransactionId** &other)

## Static Public Attributes

- static const unsigned char **ID\_TRANSACTIONID** = 0

### 6.821.1 Member Typedef Documentation

6.821.1.1 `typedef decaf::lang::PointerComparator<TransactionId>  
activemq::commands::TransactionId::COMPARATOR`

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2308), and **activemq::commands::XATransactionId** (p. 3961).

### 6.821.2 Constructor & Destructor Documentation

6.821.2.1 `activemq::commands::TransactionId::TransactionId ( )`

6.821.2.2 `activemq::commands::TransactionId::TransactionId ( const TransactionId & other )`

6.821.2.3 `virtual activemq::commands::TransactionId::~~TransactionId ( ) [virtual]`

### 6.821.3 Member Function Documentation

6.821.3.1 `virtual TransactionId* activemq::commands::TransactionId::cloneDataStructure ( ) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

#### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2308), and **activemq::commands::XATransactionId** (p. 3962).

6.821.3.2 `virtual int activemq::commands::TransactionId::compareTo ( const TransactionId & value ) const [virtual]`

6.821.3.3 `virtual void activemq::commands::TransactionId::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Implements **activemq::commands::DataStructure** (p. 1629).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2308), and **activemq::commands::XATransactionId** (p. 3962).

6.821.3.4 `virtual bool activemq::commands::TransactionId::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1630).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2309), and **activemq::commands::XATransactionId** (p. 3962).

6.821.3.5 virtual bool activemq::commands::TransactionId::equals ( const TransactionId & value ) const [virtual]

6.821.3.6 virtual unsigned char activemq::commands::TransactionId::getDataStructureType ( ) const [virtual]

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataSet** (p. 1628) type copy.

Implements **activemq::commands::DataSet** (p. 1631).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2309), and **activemq::commands::XATransactionId** (p. 3963).

6.821.3.7 virtual bool activemq::commands::TransactionId::operator< ( const TransactionId & value ) const [virtual]

6.821.3.8 TransactionId& activemq::commands::TransactionId::operator= ( const TransactionId & other )

6.821.3.9 virtual bool activemq::commands::TransactionId::operator== ( const TransactionId & value ) const [virtual]

6.821.3.10 virtual std::string activemq::commands::TransactionId::toString ( ) const [virtual]

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 796).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 2310), and **activemq::commands::XATransactionId** (p. 3964).

### 6.821.4 Field Documentation

6.821.4.1 const unsigned char activemq::commands::TransactionId::ID\_TRANSACTIONID = 0 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionId.h**

## 6.822 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3763).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMa
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.822.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3763).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.822.2 Constructor & Destructor Documentation

## 6.822 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller

### Class Reference 3777

- 6.822.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::TransactionIdMarshaller`  
( ) [inline]
- 6.822.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::~~TransactionIdMarshaller`  
( ) [inline, virtual]

### 6.822.3 Member Function Documentation

- 6.822.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseMarshal`  
( `OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut` ) throw ( `decaf::io::IOException` ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1591).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 2324), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3986).

- 6.822.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseUnmarshal`  
( `OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn` ) throw ( `decaf::io::IOException` ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2324), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 3986).

```
6.822.3.3  virtual int activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2325), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 3987).

```
6.822.3.4  virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.



## Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2325), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 3987).

6.822.3.5 virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightUnmarshal ( **OpenWireFormat** \* *wireFormat*, **commands::DataStructure** \* *dataStructure*, **decaf::io::DataInputStream** \* *dataIn*, **utils::BooleanStream** \* *bs* ) throw ( **decaf::io::IOException** ) [virtual]

Un-marshal an object instance from the data input stream.

## Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

## Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller** (p. 2326), and **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller** (p. 3988).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**TransactionIdMarshaller.h**

## 6.823 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3766).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller:

## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.823.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3766).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.823.2 Constructor & Destructor Documentation

- 6.823.2.1 **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::TransactionIdMarshaller** ( ) [*inline*]
- 6.823.2.2 **virtual activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::~~TransactionIdMarshaller** ( ) [*inline, virtual*]

### 6.823.3 Member Function Documentation

- 6.823.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseMarshal** ( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [*virtual*]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2332), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3978).

6.823.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )`  
 [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2332), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3978).

6.823.3.3 `virtual int activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2333), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3979).

```
6.823.3.4  virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2333), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3979).

```
6.823.3.5  virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

## 6.824 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller

### Class Reference 3783

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 2334), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3980).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**TransactionIdMarshaller.h**

## 6.824 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3770).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller:

#### Public Member Functions

- **TransactionIdMarshaller ()**
- virtual **~TransactionIdMarshaller ()**
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)**  
throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )**  
*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.824.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3770).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.824.2 Constructor & Destructor Documentation

6.824.2.1 **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::TransactionIdMarshaller**  
 ( ) [inline]

6.824.2.2 **virtual activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::~~TransactionIdMarshaller**  
 ( ) [inline, virtual]

### 6.824.3 Member Function Documentation

6.824.3.1 **virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseMarshal**  
 ( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2316), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

(p. 3970).

6.824.3.2 virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataInputStream \* *dataIn* ) throw ( decaf::io::IOException )  
 [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**  
 (p. 2316), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**  
 (p. 3970).

6.824.3.3 virtual int activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal1  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException )  
 [virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2317), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3971).

```
6.824.3.4  virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2317), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3971).

```
6.824.3.5  virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).



## 6.825 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller

### Class Reference 3787

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 2318), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3972).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**TransactionIdMarshaller.h**

## 6.825 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3774).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.825.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3774).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.825.2 Constructor & Destructor Documentation

6.825.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::TransactionIdMarshaller ( ) [inline]`

6.825.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::~~TransactionIdMarshaller ( ) [inline, virtual]`

### 6.825.3 Member Function Documentation

6.825.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2320), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3982).

6.825.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2320), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3982).

```
6.825.3.3 virtual int activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2321), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3983).

```
6.825.3.4 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2321), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3983).

```
6.825.3.5  virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 2322), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3984).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**TransactionIdMarshaller.h**

## 6.826 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3778).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller:

### Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

### 6.826.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3778).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.826.2 Constructor & Destructor Documentation

6.826.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::TransactionIdMarshaller`  
( ) [inline]

6.826.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::~~TransactionIdMarshaller`  
( ) [inline, virtual]

### 6.826.3 Member Function Documentation

6.826.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseMarshal`  
( `OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut` ) throw ( `decaf::io::IOException` ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1591).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 2328), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3974).

6.826.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseUnmarshal`  
( `OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn` ) throw ( `decaf::io::IOException` )  
[virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.826 activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller

### Class Reference 3793

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2328), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3974).

```
6.826.3.3 virtual int activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2329), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3975).

```
6.826.3.4 virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2329), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3975).

```
6.826.3.5  virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 2330), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3976).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**TransactionIdMarshaller.h**

## 6.827 **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3781).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMa
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**:



## Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )  
*Write a object instance to data output stream.*

## 6.827.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3781).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.827.2 Constructor &amp; Destructor Documentation

6.827.2.1 **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::TransactionIdMarshaller**  
 ( ) [inline]

6.827.2.2 **virtual activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::~~TransactionIdMarshaller**  
 ( ) [inline, virtual]

## 6.827.3 Member Function Documentation

6.827.3.1 **virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseMarshal**  
 ( **OpenWireFormat** \* wireFormat, **commands::DataStructure** \* dataStructure, **decaf::io::DataOutputStream** \* dataOut ) throw ( **decaf::io::IOException** ) [virtual]

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2312), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3966).

```
6.827.3.2  virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2312), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3966).

```
6.827.3.3  virtual int activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
-------------------	---

<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Returns

int value indicating the size of the marshaled object.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2313), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3967).

```
6.827.3.4 virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2313), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3967).

```
6.827.3.5 virtual void activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

Reimplemented in **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller** (p. 2314), and **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller** (p. 3968).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**TransactionIdMarshaller.h**

**6.828 activemq::commands::TransactionInfo Class Reference**

```
#include <src/main/activemq/commands/TransactionInfo.h>
```

Inheritance diagram for **activemq::commands::TransactionInfo**:

**Public Member Functions**

- **TransactionInfo** ()
- virtual **~TransactionInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **TransactionInfo** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **isTransactionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor \*visitor) throw ( exceptions::ActiveMQException )

*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*

### Static Public Attributes

- static const unsigned char **ID\_TRANSACTIONINFO** = 7

### Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**

## 6.828.1 Constructor & Destructor Documentation

6.828.1.1 **activemq::commands::TransactionInfo::TransactionInfo** ( )

6.828.1.2 **virtual activemq::commands::TransactionInfo::~TransactionInfo** ( ) [virtual]

## 6.828.2 Member Function Documentation

6.828.2.1 **virtual TransactionInfo\*** **activemq::commands::TransactionInfo::cloneDataStructure** ( ) const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

6.828.2.2 `virtual void activemq::commands::TransactionInfo::copyDataStructure ( const DataStructure * src ) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

#### Parameters

<i>src</i>	- Source Object
------------	-----------------

Reimplemented from `activemq::commands::BaseCommand` (p. 724).

6.828.2.3 `virtual bool activemq::commands::TransactionInfo::equals ( const DataStructure * value ) const [virtual]`

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

#### Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 725).

6.828.2.4 `virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ( ) const [virtual]`

6.828.2.5 `virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ( ) [virtual]`

6.828.2.6 `virtual unsigned char activemq::commands::TransactionInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

#### Returns

new **DataStructure** (p. 1628) type copy.

Implements `activemq::commands::DataStructure` (p. 1631).

6.828.2.7 `virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ( ) const [virtual]`

6.828.2.8 virtual **Pointer**<**TransactionId**>& **activemq::commands::TransactionInfo::getTransactionId** ( )  
[virtual]

6.828.2.9 virtual unsigned char **activemq::commands::TransactionInfo::getType** ( ) const  
[virtual]

6.828.2.10 virtual bool **activemq::commands::TransactionInfo::isTransactionInfo** ( ) const  
[inline, virtual]

### Returns

an answer of true to the **isTransactionInfo()** (p. 3788) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 728).

6.828.2.11 virtual void **activemq::commands::TransactionInfo::setConnectionId** ( const  
**Pointer**< **ConnectionId** > & *connectionId* ) [virtual]

6.828.2.12 virtual void **activemq::commands::TransactionInfo::setTransactionId** ( const  
**Pointer**< **TransactionId** > & *transactionId* ) [virtual]

6.828.2.13 virtual void **activemq::commands::TransactionInfo::setType** ( unsigned char *type* )  
[virtual]

6.828.2.14 virtual std::string **activemq::commands::TransactionInfo::toString** ( ) const  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.828.2.15 virtual **Pointer**<**Command**> **activemq::commands::TransactionInfo::visit**  
( **activemq::state::CommandVisitor** \* *visitor* ) throw (  
**exceptions::ActiveMQException** ) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.828.3 Field Documentation

- 6.828.3.1 **Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId**  
[protected]
- 6.828.3.2 **const unsigned char activemq::commands::TransactionInfo::ID\_TRANSACTIONINFO = 7** [static]
- 6.828.3.3 **Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId**  
[protected]
- 6.828.3.4 **unsigned char activemq::commands::TransactionInfo::type**  
[protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionInfo.h**

## 6.829 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3789).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller:

### Public Member Functions

- **TransactionInfoMarshaller ()**
- virtual **~TransactionInfoMarshaller ()**
- virtual **commands::DataStructure \* createObject ()** const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType ()** const  
*Get the Data Structure Type that identifies this Marshaller.*
- virtual void **tightUnmarshal (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1 (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, utils::BooleanStream \*bs)** throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*



- virtual void **tightMarshal2** (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut, utils::BooleanStream \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataInputStream \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (OpenWireFormat \*wireFormat, commands::DataStructure \*dataStructure, decaf::io::DataOutputStream \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.829.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3789).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.829.2 Constructor & Destructor Documentation

6.829.2.1 **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::TransactionInfoMarshaller**  
 ( ) [inline]

6.829.2.2 **virtual activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.829.3 Member Function Documentation

6.829.3.1 **virtual commands::DataStructure\* ac-**  
**tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::createObject (**  
**) const** [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.829.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::getDataStructureType**  
**( ) const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

**Returns**

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.829.3.3 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
  decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 751).

```
6.829.3.4 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 752).

## 6.829 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller

### Class Reference 3805

6.829.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal1 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )`  
[virtual]

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 754).

6.829.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException )` [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 755).

```
6.829.3.7 virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 756).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**TransactionInfoMarshaller.h**

## 6.830 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3793).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**:

#### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.830.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3793).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.830.2 Constructor & Destructor Documentation

6.830.2.1 **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::TransactionInfoMarshaller**  
 ( ) [inline]

6.830.2.2 **virtual activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.830.3 Member Function Documentation

6.830.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.830.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.830.3.3  virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 745).

```
6.830.3.4  virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.830 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller

### Class Reference 3809

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 746).

```
6.830.3.5  virtual int activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 747).

```
6.830.3.6  virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 748).

```
6.830.3.7 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 749).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**TransactionInfoMarshaller.h**

## 6.831 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3797).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**:

#### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )



*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.831.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3797).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.831.2 Constructor & Destructor Documentation

6.831.2.1 **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::TransactionInfoMarshaller**  
( ) [inline]

6.831.2.2 **virtual activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
( ) [inline, virtual]

### 6.831.3 Member Function Documentation

6.831.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.831.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.831.3.3  virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 731).

```
6.831.3.4  virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.831 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller

### Class Reference 3813

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 732).

```
6.831.3.5  virtual int activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 733).

```
6.831.3.6  virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
            ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 734).

```
6.831.3.7 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 736).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**TransactionInfoMarshaller.h**

## 6.832 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3801).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller**:

#### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \* wireFormat**, **commands::DataStructure \* dataStructure**, **decaf::io::DataInputStream \* dataIn**, **utils::BooleanStream \* bs**)  
**throw ( decaf::io::IOException )**

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.832.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3801).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.832.2 Constructor & Destructor Documentation

6.832.2.1 **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::TransactionInfoMarshaller**  
 ( ) [inline]

6.832.2.2 **virtual activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.832.3 Member Function Documentation

6.832.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.832.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.832.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 758).

6.832.3.4 `virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::looseUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException ) [virtual]`

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.832 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller

### Class Reference 3817

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 759).

```
6.832.3.5  virtual int activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 760).

```
6.832.3.6  virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 762).

```
6.832.3.7 virtual void activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller** (p. 763).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**TransactionInfoMarshaller.h**

## 6.833 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3805).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**:

#### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )



*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.833.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3805).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.833.2 Constructor & Destructor Documentation

6.833.2.1 **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::TransactionInfoMarshaller**  
 ( ) [inline]

6.833.2.2 **virtual activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
 ( ) [inline, virtual]

### 6.833.3 Member Function Documentation

6.833.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.833.3.2  virtual unsigned char activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.833.3.3  virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 738).

```
6.833.3.4  virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

### 6.833 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller

#### Class Reference 3821

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 739).

```
6.833.3.5  virtual int activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 740).

```
6.833.3.6  virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 741).

```
6.833.3.7 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 742).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**TransactionInfoMarshaller.h**

## 6.834 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3809).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**:

#### Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )

*Un-marshal an object instance from the data input stream.*

- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write the booleans that this object uses to a BooleanStream.*

- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( **decaf::io::IOException** )

*Un-marshal an object instance from the data input stream.*

- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( **decaf::io::IOException** )

*Write a object instance to data output stream.*

### 6.834.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3809).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

### 6.834.2 Constructor & Destructor Documentation

6.834.2.1 **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::TransactionInfoMarshaller**  
( ) [inline]

6.834.2.2 **virtual activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::~~TransactionInfoMarshaller**  
( ) [inline, virtual]

### 6.834.3 Member Function Documentation

6.834.3.1 **virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::createObject** ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

```
6.834.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::getDataStructureType
( ) const [virtual]
```

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

```
6.834.3.3 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 765).

```
6.834.3.4 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

## 6.834 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller

### Class Reference 3825

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 766).

```
6.834.3.5  virtual int activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 767).

```
6.834.3.6  virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 768).

```
6.834.3.7 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h**

## 6.835 activemq::state::TransactionState Class Reference

```
#include <src/main/activemq/state/TransactionState.h>
```

#### Public Member Functions

- **TransactionState** (const **Pointer**< **TransactionId** > &id)
- virtual ~**TransactionState** ()
- std::string **toString** () const
- void **addCommand** (const **Pointer**< **Command** > &operation)
- void **checkShutdown** () const
- void **shutdown** ()
- const **StlList**< **Pointer**< **Command** > > & **getCommands** () const
- const **Pointer**< **TransactionId** > & **getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const
- void **addProducerState** (const **Pointer**< **ProducerState** > &producerState)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** ()



### 6.835.1 Constructor & Destructor Documentation

6.835.1.1 `activemq::state::TransactionState::TransactionState ( const Pointer< TransactionId > & id )`

6.835.1.2 `virtual activemq::state::TransactionState::~~TransactionState ( ) [virtual]`

### 6.835.2 Member Function Documentation

6.835.2.1 `void activemq::state::TransactionState::addCommand ( const Pointer< Command > & operation )`

6.835.2.2 `void activemq::state::TransactionState::addProducerState ( const Pointer< ProducerState > & producerState )`

6.835.2.3 `void activemq::state::TransactionState::checkShutdown ( ) const`

6.835.2.4 `const StlList< Pointer<Command> >& activemq::state::TransactionState::getCommands ( ) const [inline]`

6.835.2.5 `const Pointer<TransactionId>& activemq::state::TransactionState::getId ( ) const [inline]`

6.835.2.6 `int activemq::state::TransactionState::getPreparedResult ( ) const [inline]`

6.835.2.7 `std::vector< Pointer<ProducerState> > activemq::state::TransactionState::getProducerStates ( )`

6.835.2.8 `bool activemq::state::TransactionState::isPrepared ( ) const [inline]`

6.835.2.9 `void activemq::state::TransactionState::setPrepared ( bool prepared ) [inline]`

6.835.2.10 `void activemq::state::TransactionState::setPreparedResult ( int preparedResult ) [inline]`

6.835.2.11 `void activemq::state::TransactionState::shutdown ( ) [inline]`

6.835.2.12 `std::string activemq::state::TransactionState::toString ( ) const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/TransactionState.h`

## 6.836 decaf::internal::util::concurrent::Transferer< E > Class Template Reference

Shared internal API for dual stacks and queues.

```
#include <src/main/decaf/internal/util/concurrent/Transferer.h>
```

Inheritance diagram for decaf::internal::util::concurrent::Transferer< E >:

### 6.836.1 Detailed Description

```
template<typename E>class decaf::internal::util::concurrent::Transferer< E >
```

Shared internal API for dual stacks and queues.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**Transferer.h**

## 6.837 decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

```
#include <src/main/decaf/internal/util/concurrent/TransferQueue.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferQueue< E >:

### Public Member Functions

- **TransferQueue** ()

*Node class for **TransferQueue** (p. 3815).*

- virtual ~**TransferQueue** ()
- virtual void **transfer** (E \*e, bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )  
*Performs a put.*
- virtual E \* **transfer** (bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )  
*Performs a take.*

### 6.837.1 Detailed Description

```
template<typename E>class decaf::internal::util::concurrent::TransferQueue< E >
```

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

The algorithm is a little simpler than that for stacks because fulfillers do not need explicit nodes, and matching is done by CAS'ing QNode.item field from non-null to null (for put) or vice versa (for take).

### 6.837.2 Constructor & Destructor Documentation

6.837.2.1 `template<typename E > decaf::internal::util::concurrent::TransferQueue< E >::TransferQueue ( ) [inline]`

Node class for **TransferQueue** (p.3815).

Tries to cancel by CAS'ing ref to NULL if that succeeds then we mark as cancelled. Returns true if this node is known to be off the queue because its next pointer has been forgotten due to an advanceHead operation.

6.837.2.2 `template<typename E > virtual decaf::internal::util::concurrent::TransferQueue< E >::~~TransferQueue ( ) [inline, virtual]`

### 6.837.3 Member Function Documentation

6.837.3.1 `template<typename E > virtual void decaf::internal::util::concurrent::TransferQueue< E >::transfer ( E * e, bool timed, long long nanos ) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Performs a put.

#### Parameters

<i>e</i>	the item to be handed to a consumer;
<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

#### Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the consumer to accept the item offered.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer**< **E** > (p. 3815).

```
6.837.3.2  template<typename E > virtual E*
            decaf::internal::util::concurrent::TransferQueue<
            E >::transfer ( bool timed, long long nanos ) throw
            ( decaf::util::concurrent::TimeoutException,
            decaf::lang::exceptions::InterruptedException ) [inline,
            virtual]
```

Performs a take.

#### Parameters

<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

#### Returns

the item provided or received;

#### Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the producer to offer an item.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer**< **E** > (p. 3815).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferQueue.h**

## 6.838 decaf::internal::util::concurrent::TransferStack< E > Class Template Reference

```
#include <src/main/decaf/internal/util/concurrent/TransferStack.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferStack< E >:

#### Public Member Functions

- **TransferStack** ()
- virtual **~TransferStack** ()
- virtual void **transfer** (E \*e, bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )  
*Performs a put.*

## 6.838 decaf::internal::util::concurrent::TransferStack< E > Class Template

### Reference

3831

- virtual E \* **transfer** (bool timed, long long nanos) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException )

*Performs a take.*

```
template<typename E> class decaf::internal::util::concurrent::TransferStack< E >
```

### 6.838.1 Constructor & Destructor Documentation

6.838.1.1 `template<typename E > decaf::internal::util::concurrent::TransferStack< E >::TransferStack ( ) [inline]`

6.838.1.2 `template<typename E > virtual decaf::internal::util::concurrent::TransferStack< E >::~~TransferStack ( ) [inline, virtual]`

### 6.838.2 Member Function Documentation

6.838.2.1 `template<typename E > virtual void decaf::internal::util::concurrent::TransferStack< E >::transfer ( E * e, bool timed, long long nanos ) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Performs a put.

#### Parameters

<i>e</i>	the item to be handed to a consumer;
<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

#### Exceptions

<i>TimeoutException</i>	if the operation timed out waiting for the consumer to accept the item offered.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements `decaf::internal::util::concurrent::Transferer< E >` (p. 3815).

6.838.2.2 `template<typename E > virtual E* decaf::internal::util::concurrent::TransferStack< E >::transfer ( bool timed, long long nanos ) throw ( decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException ) [inline, virtual]`

Performs a take.

**Parameters**

<i>timed</i>	if this operation should timeout
<i>nanos</i>	the timeout, in nanoseconds

**Returns**

the item provided or received;

**Exceptions**

<i>TimeoutException</i>	if the operation timed out waiting for the producer to offer an item.
<i>InterruptedException</i>	if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3815).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**TransferStack.h**

**6.839 activemq::transport::Transport Class Reference**

Interface for a transport layer for command objects.

```
#include <src/main/activemq/transport/Transport.h>
```

Inheritance diagram for activemq::transport::Transport:

**Public Member Functions**

- virtual **~Transport** ()
- virtual void **start** ()=0 throw ( decaf::io::IOException )  
*Starts the **Transport** (p. 3819), the send methods of a **Transport** (p. 3819) will throw an exception if used before the **Transport** (p. 3819) is started.*
- virtual void **stop** ()=0 throw ( decaf::io::IOException )  
*Stops the **Transport** (p. 3819).*
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given command to the broker and then waits for the response.*

- virtual **Pointer< Response > request** (const **Pointer< Command >** &command, unsigned int timeout)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends the given command to the broker and then waits for the response.*
- virtual void **setWireFormat** (const **Pointer< wireformat::WireFormat >** &wireFormat)=0  
*Sets the WireFormat instance to use.*
- virtual void **setTransportListener** (**TransportListener** \*listener)=0  
*Sets the observer of asynchronous events from this transport.*
- virtual **TransportListener** \* **getTransportListener** () const =0  
*Gets the observer of asynchronous events from this transport.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeid)=0  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const =0  
*Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const =0  
*Is the **Transport** (p. 3819) Connected to its Broker.*
- virtual bool **isClosed** () const =0  
*Has the **Transport** (p. 3819) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const =0
- virtual void **reconnect** (const **decaf::net::URI** &uri)=0 throw ( decaf::io::IOException )  
*reconnect to another location*

### 6.839.1 Detailed Description

Interface for a transport layer for command objects.

Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 3819) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 3819) layer runs. Transports should be given an instance of a WireFormat object when created so that they can turn the built in Commands to / from the required wire format encoding.

### 6.839.2 Constructor & Destructor Documentation

6.839.2.1 virtual activemq::transport::Transport::~Transport ( ) [inline, virtual]

### 6.839.3 Member Function Documentation

6.839.3.1 `virtual std::string activemq::transport::Transport::getRemoteAddress ( ) const`  
`[pure virtual]`

#### Returns

the remote address for this connection

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1839), `activemq::transport::IOTransport` (p. 2108), `activemq::transport::mock::MockTransport` (p. 2727), and `activemq::transport::TransportFilter` (p. 3830).

6.839.3.2 `virtual TransportListener* activemq::transport::Transport::getTransportListener ( ) const`  
`[pure virtual]`

Gets the observer of asynchronous events from this transport.

#### Returns

the listener of transport events.

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1839), `activemq::transport::IOTransport` (p. 2108), `activemq::transport::mock::MockTransport` (p. 2728), and `activemq::transport::TransportFilter` (p. 3830).

6.839.3.3 `virtual bool activemq::transport::Transport::isClosed ( ) const`  
`[pure virtual]`

Has the **Transport** (p. 3819) been shutdown and no longer usable.

#### Returns

true if the **Transport** (p. 3819)

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1840), `activemq::transport::IOTransport` (p. 2108), `activemq::transport::mock::MockTransport` (p. 2728), `activemq::transport::tcp::TcpTransport` (p. 3698), and `activemq::transport::TransportFilter` (p. 3830).

6.839.3.4 `virtual bool activemq::transport::Transport::isConnected ( ) const`  
`[pure virtual]`

Is the **Transport** (p. 3819) Connected to its Broker.

#### Returns

true if a connection has been made.

Implemented in `activemq::transport::failover::FailoverTransport` (p. 1840), `activemq::transport::IOTransport` (p. 2108), `activemq::transport::mock::MockTransport` (p. 2728), `activemq::transport::tcp::TcpTransport` (p. 3699), and `activemq::transport::TransportFilter` (p. 3831).



6.839.3.5 `virtual bool activemq::transport::Transport::isFaultTolerant ( ) const [pure virtual]`

Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.

#### Returns

true if the **Transport** (p. 3819) is fault tolerant.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1840), **activemq::transport::IOTransport** (p. 2109), **activemq::transport::mock::MockTransport** (p. 2729), **activemq::transport::tcp::TcpTransport** (p. 3699), and **activemq::transport::TransportFilter** (p. 3831).

Referenced by **activemq::transport::TransportFilter::isFaultTolerant()**.

6.839.3.6 `virtual Transport* activemq::transport::Transport::narrow ( const std::type_info & typeId ) [pure virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.

#### Parameters

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

#### Returns

the requested Object. or NULL if its not in this chain.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1841), **activemq::transport::IOTransport** (p. 2109), **activemq::transport::mock::MockTransport** (p. 2729), and **activemq::transport::TransportFilter** (p. 3831).

Referenced by **activemq::transport::failover::FailoverTransport::narrow()**.

6.839.3.7 `virtual void activemq::transport::Transport::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
--------------------	---

<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.
--------------------------------------	--

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3234), **activemq::transport::failover::FailoverTransport** (p. 1842), **activemq::transport::inactivity::InactivityMonitor** (p. 1966), **activemq::transport::IOTransport** (p. 2109), **activemq::transport::logging::LoggingTransport** (p. 2362), **activemq::transport::mock::MockTransport** (p. 2730), **activemq::transport::TransportFilter** (p. 3832), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2852).

6.839.3.8 **virtual void activemq::transport::Transport::reconnect ( const decaf::net::URI & uri ) throw ( decaf::io::IOException )** [pure virtual]

reconnect to another location

#### Parameters

<i>uri</i>
------------

#### Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1842), and **activemq::transport::TransportFilter** (p. 3833).

6.839.3.9 **virtual Pointer<Response> activemq::transport::Transport::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )** [pure virtual]

Sends the given command to the broker and then waits for the response.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Returns

the response from the broker.

#### Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3235), **activemq::transport::failover::FailoverTransport** (p. 1843), **activemq::transport::IOTransport** (p. 2110), **activemq::transport::logging::LoggingTransport** (p. 2362), **activemq::transport::mock::MockTransport** (p. 2730), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2852).

(p. 2731), **activemq::transport::TransportFilter** (p. 3833), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2853).

```
6.839.3.10 virtual Pointer<Response> activemq::transport::Transport::request
( const Pointer< Command > & command, unsigned
  int timeout ) throw ( decaf::io::IOException,
  decaf::lang::exceptions::UnsupportedOperationException ) [pure
  virtual]
```

Sends the given command to the broker and then waits for the response.

#### Parameters

<i>command</i>	- The command to be sent.
<i>timeout</i>	- The time to wait for this response.

#### Returns

the response from the broker.

#### Exceptions

<i>IOException</i>	if an exception occurs during the read of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3235), **activemq::transport::failover::FailoverTransport** (p. 1843), **activemq::transport::io::IOTransport** (p. 2110), **activemq::transport::logging::LoggingTransport** (p. 2363), **activemq::transport::mock::MockTransport** (p. 2730), **activemq::transport::TransportFilter** (p. 3833), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2854).

```
6.839.3.11 virtual void activemq::transport::Transport::setTransportListener (
  TransportListener * listener ) [pure virtual]
```

Sets the observer of asynchronous events from this transport.

#### Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1845), **activemq::transport::io::IOTransport** (p. 2111), **activemq::transport::mock::MockTransport** (p. 2732), and **activemq::transport::TransportFilter** (p. 3834).

```
6.839.3.12 virtual void activemq::transport::Transport::setWireFormat ( const Pointer<
  wireformat::WireFormat > & wireFormat ) [pure virtual]
```

Sets the WireFormat instance to use.

**Parameters**

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implemented in **activemq::transport::IOTransport** (p. 2112), and **activemq::transport::TransportFilter** (p. 3834).

6.839.3.13 `virtual void activemq::transport::Transport::start ( ) throw ( decaf::io::IOException )` [pure virtual]

Starts the **Transport** (p. 3819), the send methods of a **Transport** (p. 3819) will throw an exception if used before the **Transport** (p. 3819) is started.

**Exceptions**

<i>IOException</i>	if and error occurs while starting the <b>Transport</b> (p. 3819).
--------------------	--

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3236), **activemq::transport::failover::FailoverTransport** (p. 1846), **activemq::transport::IOTransport** (p. 2112), **activemq::transport::mock::MockTransport** (p. 2733), **activemq::transport::TransportFilter** (p. 3834), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2854).

6.839.3.14 `virtual void activemq::transport::Transport::stop ( ) throw ( decaf::io::IOException )` [pure virtual]

Stops the **Transport** (p. 3819).

**Exceptions**

<i>IOException</i>	if an error occurs while stopping the transport.
--------------------	--

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1846), **activemq::transport::IOTransport** (p. 2112), **activemq::transport::mock::MockTransport** (p. 2733), and **activemq::transport::TransportFilter** (p. 3835).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/Transport.h`

**6.840 activemq::transport::TransportFactory Class Reference**

Defines the interface for Factories that create Transports or TransportFilters.

```
#include <src/main/activemq/transport/TransportFactory.h>
```

Inheritance diagram for **activemq::transport::TransportFactory**:

## Public Member Functions

- virtual `~TransportFactory()`
- virtual `Pointer<Transport> create (const decaf::net::URI &location)=0` throw ( exceptions::ActiveMQException )  
*Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.*
- virtual `Pointer<Transport> createComposite (const decaf::net::URI &location)=0` throw ( exceptions::ActiveMQException )  
*Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.*

### 6.840.1 Detailed Description

Defines the interface for Factories that create Transports or TransportFilters.

The factory should be able to create either a completely configured **Transport** (p. 3819) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimmed down version that is used in composite transports like Failover or Fanout.

Since

3.0

### 6.840.2 Constructor & Destructor Documentation

6.840.2.1 `virtual activemq::transport::TransportFactory::~TransportFactory ( ) [inline, virtual]`

### 6.840.3 Member Function Documentation

6.840.3.1 `virtual Pointer<Transport> activemq::transport::TransportFactory::create ( const decaf::net::URI & location ) throw ( exceptions::ActiveMQException ) [pure virtual]`

Creates a fully configured **Transport** (p. 3819) instance which could be a chain of filters and transports.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implemented in `activemq::transport::failover::FailoverTransportFactory` (p. 1847), `activemq::transport::mock::MockTransportFactory` (p. 2735), and `activemq::transport::tcp::TcpTransportFactory` (p. 3700).

6.840.3.2 virtual **Pointer**<**Transport**> **activemq::transport::TransportFactory::createComposite** ( const **decaf::net::URI** & *location* ) throw ( **exceptions::ActiveMQException** )  
[pure virtual]

Creates a slimmed down **Transport** (p. 3819) instance which can be used in composite transport instances.

#### Parameters

<i>location</i>	- URI location to connect to plus any properties to assign.
-----------------	---

#### Exceptions

<i>ActiveMQException</i>	if an error occurs
--------------------------	--------------------

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1848), **activemq::transport::mock::MockTransportFactory** (p. 2735), and **activemq::transport::tcp::TcpTransportFactory** (p. 3701).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportFactory.h**

## 6.841 **activemq::transport::TransportFilter** Class Reference

A filter on the transport layer.

```
#include <src/main/activemq/transport/TransportFilter.h>
```

Inheritance diagram for **activemq::transport::TransportFilter**:

#### Public Member Functions

- **TransportFilter** (const **Pointer**< **Transport** > &next)  
*Constructor.*
- virtual ~**TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)  
*Event handler for an exception from a command transport.*
- virtual void **transportInterrupted** ()  
*The transport has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()  
*The transport has resumed after an interruption.*

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Sends a one-way command.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Not supported by this class - throws an exception.*
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException )  
*Not supported by this class - throws an exception.*
- virtual void **setTransportListener** (**TransportListener** \*listener)  
*Sets the observer of asynchronous exceptions from this transport.*
- virtual **TransportListener** \* **getTransportListener** () const  
*Gets the observer of asynchronous exceptions from this transport.*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)  
*Sets the WireFormat instance to use.*
- virtual void **start** () throw ( decaf::io::IOException )  
*Starts this transport object and creates the thread for polling on the input stream for commands.*
- virtual void **stop** () throw ( decaf::io::IOException )  
*Stops the **Transport** (p. 3819).*
- virtual void **close** () throw ( decaf::io::IOException )  
*Stops the polling thread and closes the streams.*
- virtual **Transport** \* **narrow** (const std::type\_info &typeid)  
*Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const  
*Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const  
*Is the **Transport** (p. 3819) Connected to its Broker.*
- virtual bool **isClosed** () const  
*Has the **Transport** (p. 3819) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw ( decaf::io::IOException )  
*reconnect to another location*

## Protected Member Functions

- void **fire** (const **decaf::lang::Exception** &ex)  
*Notify the listener of the thrown Exception.*
- void **fire** (const **Pointer**< **Command** > &command)  
*Notify the listener of the new incoming Command.*

## Protected Attributes

- **Pointer< Transport > next**  
*The transport that this filter wraps around.*
- **TransportListener \* listener**  
*Listener of this transport.*

### 6.841.1 Detailed Description

A filter on the transport layer.

**Transport** (p. 3819) filters implement the **Transport** (p. 3819) interface and optionally delegate calls to another **Transport** (p. 3819) object.

#### Since

1.0

### 6.841.2 Constructor & Destructor Documentation

6.841.2.1 `activemq::transport::TransportFilter::TransportFilter ( const Pointer< Transport > & next )`

Constructor.

#### Parameters

<code>next</code>	- the next <b>Transport</b> (p. 3819) in the chain
-------------------	--

6.841.2.2 `virtual activemq::transport::TransportFilter::~~TransportFilter ( ) [inline, virtual]`

### 6.841.3 Member Function Documentation

6.841.3.1 `virtual void activemq::transport::TransportFilter::close ( ) throw ( decaf::io::IOException ) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

#### Exceptions

<code>IOException</code>	if an error occurs while closing the <b>Transport</b> (p. 3819).
--------------------------	--

Implements **decaf::io::Closeable** (p. 1121).



Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3234), **activemq::transport::inactivity::InactivityMonitor** (p. 1965), **activemq::transport::tcp::TcpTransport** (p. 3697), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2852).

6.841.3.2 `void activemq::transport::TransportFilter::fire ( const Pointer< Command > & command ) [protected]`

Notify the listener of the new incoming Command.

#### Parameters

<code>command</code>	- the command to send to the listener
----------------------	---------------------------------------

6.841.3.3 `void activemq::transport::TransportFilter::fire ( const decaf::lang::Exception & ex ) [protected]`

Notify the listener of the thrown Exception.

#### Parameters

<code>ex</code>	- the exception to send to listeners
-----------------	--------------------------------------

6.841.3.4 `virtual std::string activemq::transport::TransportFilter::getRemoteAddress ( ) const [inline, virtual]`

#### Returns

the remote address for this connection

Implements **activemq::transport::Transport** (p. 3821).

6.841.3.5 `virtual TransportListener* activemq::transport::TransportFilter::getTransportListener ( ) const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

#### Returns

The listener of transport events.

Implements **activemq::transport::Transport** (p. 3821).

6.841.3.6 `virtual bool activemq::transport::TransportFilter::isClosed ( ) const [inline, virtual]`

Has the **Transport** (p. 3819) been shutdown and no longer usable.

**Returns**

true if the **Transport** (p. 3819)

Implements **activemq::transport::Transport** (p. 3821).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3698).

6.841.3.7 `virtual bool activemq::transport::TransportFilter::isConnected ( ) const`  
`[inline, virtual]`

Is the **Transport** (p. 3819) Connected to its Broker.

**Returns**

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3821).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3699).

6.841.3.8 `virtual bool activemq::transport::TransportFilter::isFaultTolerant ( ) const`  
`[inline, virtual]`

Is this **Transport** (p. 3819) fault tolerant, meaning that it will reconnect to a broker on disconnect.

**Returns**

true if the **Transport** (p. 3819) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3822).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 3699).

References `activemq::transport::Transport::isFaultTolerant()`.

6.841.3.9 `virtual Transport* activemq::transport::TransportFilter::narrow ( const`  
`std::type_info & typeId ) [virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3819) to allow a higher level transport to skip intermediate Transports in certain circumstances.

**Parameters**

<i>typeId</i>	- The type_info of the Object we are searching for.
---------------	---

**Returns**

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3822).

6.841.3.10 `virtual void activemq::transport::TransportFilter::onCommand ( const Pointer< Command > & command ) [virtual]`

Event handler for the receipt of a command.

#### Parameters

<i>command</i>	- the received command object.
----------------	--------------------------------

Implements **activemq::transport::TransportListener** (p. 3836).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3234), **activemq::transport::inactivity::InactivityMonitor** (p. 1966), **activemq::transport::logging::LoggingTransport** (p. 2361), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2852).

6.841.3.11 `virtual void activemq::transport::TransportFilter::oneway ( const Pointer< Command > & command ) throw ( decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException ) [inline, virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

#### Parameters

<i>command</i>	the command to be sent.
----------------	-------------------------

#### Exceptions

<i>IOException</i>	if an exception occurs during writing of the command.
<i>UnsupportedOperationException</i>	if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3822).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3234), **activemq::transport::inactivity::InactivityMonitor** (p. 1966), **activemq::transport::logging::LoggingTransport** (p. 2362), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2852).

6.841.3.12 `virtual void activemq::transport::TransportFilter::onException ( const decaf::lang::Exception & ex ) [virtual]`

Event handler for an exception from a command transport.

#### Parameters

<i>ex</i>	The exception to handle.
-----------	--------------------------

Implements **activemq::transport::TransportListener** (p. 3837).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 1967).

6.841.3.13 `virtual void activemq::transport::TransportFilter::reconnect ( const  
decaf::net::URI & uri ) throw ( decaf::io::IOException ) [virtual]`

reconnect to another location

#### Parameters

<i>uri</i>	
------------	--

#### Exceptions

<i>IOException</i>	on failure of if not supported
--------------------	--------------------------------

Implements **activemq::transport::Transport** (p. 3823).

6.841.3.14 `virtual Pointer<Response> activemq::transport::TransportFilter::request  
( const Pointer< Command > & command, unsigned  
int timeout ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Not supported by this class - throws an exception.

#### Parameters

<i>command</i>	- The command that is sent as a request
<i>timeout</i>	- The the time to wait for a response.

#### Exceptions

<i>IOException</i>	
<i>UnsupportedOperation- Exception.</i>	

Implements **activemq::transport::Transport** (p. 3824).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3235),  
**activemq::transport::logging::LoggingTransport** (p. 2363), and **activemq::wireformat::openwire::OpenWire**  
(p. 2854).

6.841.3.15 `virtual Pointer<Response> activemq::transport::TransportFilter::request ( const Pointer< Command > & command ) throw ( decaf::io::IOException,  
decaf::lang::exceptions::UnsupportedOperationException )  
[inline, virtual]`

Not supported by this class - throws an exception.

### Parameters

<i>command</i>	the command that is sent as a request
----------------	---------------------------------------

### Exceptions

<i>IOException</i>	
<i>UnsupportedOperationException</i>	

Implements **activemq::transport::Transport** (p. 3823).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3235), **activemq::transport::logging::LoggingTransport** (p. 2362), and **activemq::wireformat::openwire::OpenWireFormatNeg** (p. 2853).

**6.841.3.16** virtual void activemq::transport::TransportFilter::setTransportListener ( **TransportListener** \* *listener* ) [inline, virtual]

Sets the observer of asynchronous exceptions from this transport.

### Parameters

<i>listener</i>	the listener of transport events.
-----------------	-----------------------------------

Implements **activemq::transport::Transport** (p. 3824).

**6.841.3.17** virtual void activemq::transport::TransportFilter::setWireFormat ( const **Pointer**< **wireformat::WireFormat** > & *wireFormat* ) [inline, virtual]

Sets the WireFormat instance to use.

### Parameters

<i>wireFormat</i>	The WireFormat the object used to encode / decode commands.
-------------------	---

Implements **activemq::transport::Transport** (p. 3824).

**6.841.3.18** virtual void activemq::transport::TransportFilter::start ( ) throw ( **decaf::io::IOException** ) [virtual]

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

### Exceptions

<i>IOException</i>	if an error occurs or if this transport has already been closed.
--------------------	--

Implements **activemq::transport::Transport** (p. 3825).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 3236),  
and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2854).

6.841.3.19 `virtual void activemq::transport::TransportFilter::stop ( ) throw (   
decaf::io::IOException ) [virtual]`

Stops the **Transport** (p. 3819).

### Exceptions

<i>IOException</i>	if an error occurs while stopping the <b>Transport</b> (p. 3819).
--------------------	---

Implements **activemq::transport::Transport** (p. 3825).

6.841.3.20 `virtual void activemq::transport::TransportFilter::transportInterrupted ( )   
[virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3837).

6.841.3.21 `virtual void activemq::transport::TransportFilter::transportResumed ( )   
[virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3837).

## 6.841.4 Field Documentation

6.841.4.1 **TransportListener\* activemq::transport::TransportFilter::listener**  
[protected]

Listener of this transport.

6.841.4.2 **Pointer<Transport> activemq::transport::TransportFilter::next**  
[protected]

The transport that this filter wraps around.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportFilter.h**

## 6.842 activemq::transport::TransportListener Class Reference

A listener of asynchronous exceptions from a command transport object.

```
#include <src/main/activemq/transport/TransportListener.h>
```

Inheritance diagram for activemq::transport::TransportListener:

### Public Member Functions

- virtual **~TransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)=0  
*Event handler for the receipt of a command.*
- virtual void **onException** (const **decaf::lang::Exception** &ex)=0  
*Event handler for an exception from a command transport.*
- virtual void **transportInterrupted** ()=0  
*The transport has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()=0  
*The transport has resumed after an interruption.*

### 6.842.1 Detailed Description

A listener of asynchronous exceptions from a command transport object.

### 6.842.2 Constructor & Destructor Documentation

6.842.2.1 virtual activemq::transport::TransportListener::~~TransportListener ( )  
[inline, virtual]

### 6.842.3 Member Function Documentation

6.842.3.1 virtual void activemq::transport::TransportListener::onCommand ( const **Pointer**< **Command** > & *command* ) [pure virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p.3819) deletes the command upon receipt.

### Parameters

<i>command</i>	the received command object.
----------------	------------------------------

Implemented in **activemq::core::ActiveMQConnection** (p.257), **activemq::transport::correlator::ResponseCorrelator**

(p. 3234), **activemq::transport::failover::FailoverTransportListener** (p. 1850), **activemq::transport::inactivity::InactivityMonitor** (p. 1966), **activemq::transport::logging::LoggingTransport** (p. 2361), **activemq::transport::mock::InternalCommandListener** (p. 2086), **activemq::transport::Transport** (p. 3832), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2852).

**6.842.3.2** `virtual void activemq::transport::TransportListener::onException ( const decaf::lang::Exception & ex ) [pure virtual]`

Event handler for an exception from a command transport.

#### Parameters

<code>ex</code>	The exception being propagated to this listener to handle.
-----------------	--

Implemented in **activemq::core::ActiveMQConnection** (p. 258), **activemq::transport::failover::BackupTransport** (p. 719), **activemq::transport::failover::FailoverTransportListener** (p. 1850), **activemq::transport::inactivity::InactivityMonitor** (p. 1967), and **activemq::transport::TransportFilter** (p. 3832).

**6.842.3.3** `virtual void activemq::transport::TransportListener::transportInterrupted ( ) [pure virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implemented in **activemq::core::ActiveMQConnection** (p. 264), **activemq::transport::DefaultTransportListener** (p. 1671), **activemq::transport::failover::FailoverTransportListener** (p. 1850), and **activemq::transport::TransportFilter** (p. 3835).

**6.842.3.4** `virtual void activemq::transport::TransportListener::transportResumed ( ) [pure virtual]`

The transport has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnection** (p. 264), **activemq::transport::DefaultTransportListener** (p. 1671), **activemq::transport::failover::FailoverTransportListener** (p. 1850), and **activemq::transport::TransportFilter** (p. 3835).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportListener.h`

## 6.843 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 3819) Factories that are available to the client at runtime.

```
#include <src/main/activemq/transport/TransportRegistry.h>
```



## Public Member Functions

- virtual **~TransportRegistry** ()
- **TransportFactory** \* **findFactory** (const std::string &name) const throw ( decaf::lang::exceptions::NoSuchElementException )
 

*Gets a Registered **TransportFactory** (p. 3825) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*
- void **registerFactory** (const std::string &name, **TransportFactory** \*factory) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )
 

*Registers a new **TransportFactory** (p. 3825) with this Registry.*
- void **unregisterFactory** (const std::string &name)
 

*Unregisters the Factory with the given name and deletes that instance of the Factory.*
- std::vector< std::string > **getTransportNames** () const
 

*Retrieves a list of the names of all the Registered Transport's in this Registry.*

## Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()
 

*Gets the single instance of the **TransportRegistry** (p. 3837).*

### 6.843.1 Detailed Description

Registry of all **Transport** (p. 3819) Factories that are available to the client at runtime.

New Transport's must have a factory registered here before a connection attempt is made.

Since

3.0

### 6.843.2 Constructor & Destructor Documentation

6.843.2.1 virtual activemq::transport::TransportRegistry::~~TransportRegistry ( )  
[virtual]

### 6.843.3 Member Function Documentation

6.843.3.1 **TransportFactory**\* activemq::transport::TransportRegistry::findFactory  
( const std::string & name ) const throw ( decaf::lang::exceptions::NoSuchElementException )

Gets a Registered **TransportFactory** (p. 3825) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

**Parameters**

<i>name</i>	The name of the Factory to find in the Registry.
-------------	--

**Returns**

the Factory registered under the given name.

**Exceptions**

<i>NoSuchElementException</i>	if no factory is registered with that name.
-------------------------------	---

6.843.3.2 **static TransportRegistry& activemq::transport::TransportRegistry::getInstance ( )**  
[static]

Gets the single instance of the **TransportRegistry** (p. 3837).

**Returns**

reference to the single instance of this Registry

6.843.3.3 **std::vector<std::string> activemq::transport::TransportRegistry::getTransportNames ( ) const**

Retrieves a list of the names of all the Registered Transport's in this Registry.

**Returns**

stl vector of strings with all the **Transport** (p. 3819) names registered.

6.843.3.4 **void activemq::transport::TransportRegistry::registerFactory (**  
**const std::string & name, TransportFactory \* factory ) throw**  
**( decaf::lang::exceptions::IllegalArgumentException,**  
**decaf::lang::exceptions::NullPointerException )**

Registers a new **TransportFactory** (p. 3825) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

**Parameters**

<i>name</i>	The name of the new Factory to register.
<i>factory</i>	The new Factory to add to the Registry.

### Exceptions

<i>IllegalArgumentException</i>	is name is the empty string.
<i>NullPointerException</i>	if the Factory is Null.

6.843.3.5 void activemq::transport::TransportRegistry::unregisterFactory ( const std::string & name )

Unregisters the Factory with the given name and deletes that instance of the Factory.

### Parameters

<i>name</i>	Name of the Factory to unregister and destroy
-------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportRegistry.h**

## 6.844 tree\_desc\_s Struct Reference

```
#include <src/main/decaf/internal/util/zip/deflate.h>
```

### Data Fields

- **ct\_data** \* dyn\_tree
- int max\_code
- static\_tree\_desc \* stat\_desc

#### 6.844.1 Field Documentation

6.844.1.1 ct\_data\* tree\_desc\_s::dyn\_tree

6.844.1.2 int tree\_desc\_s::max\_code

6.844.1.3 static\_tree\_desc\* tree\_desc\_s::stat\_desc

The documentation for this struct was generated from the following file:

- src/main/decaf/internal/util/zip/**deflate.h**

## 6.845 decaf::lang::Thread::UncaughtExceptionHandler Class Reference

Interface for handlers invoked when a **Thread** (p.3707) abruptly terminates due to an uncaught exception.

```
#include <src/main/decaf/lang/Thread.h>
```

### Public Member Functions

- virtual **~UncaughtExceptionHandler** ()
- virtual void **uncaughtException** (const **Thread** \*thread, const **Throwable** &error)=0 throw ()

*Method invoked when the given thread terminates due to the given uncaught exception.*

#### 6.845.1 Detailed Description

Interface for handlers invoked when a **Thread** (p.3707) abruptly terminates due to an uncaught exception.

#### 6.845.2 Constructor & Destructor Documentation

6.845.2.1 virtual decaf::lang::Thread::UncaughtExceptionHandler::~~UncaughtExceptionHandler ( ) [inline, virtual]

#### 6.845.3 Member Function Documentation

6.845.3.1 virtual void decaf::lang::Thread::UncaughtExceptionHandler::uncaughtException ( const **Thread** \* thread, const **Throwable** & error ) throw () [pure virtual]

Method invoked when the given thread terminates due to the given uncaught exception.

This method is defined to indicate that it will not throw an exception, throwing and exception from this method will on most systems result in a segmentation fault.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Thread.h**

## 6.846 decaf::net::UnknownHostException Class Reference

```
#include <src/main/decaf/net/UnknownHostException.h>
```

Inheritance diagram for decaf::net::UnknownHostException:

## Public Member Functions

- **UnknownHostException** () throw ()  
*Default Constructor.*
- **UnknownHostException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **UnknownHostException** (const **UnknownHostException** &ex) throw ()  
*Copy Constructor.*
- **UnknownHostException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnknownHostException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnknownHostException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **UnknownHostException** \* clone () const  
*Clones this exception.*
- virtual ~**UnknownHostException** () throw ()

## 6.846.1 Constructor & Destructor Documentation

6.846.1.1 decaf::net::UnknownHostException::UnknownHostException ( ) throw ()  
[inline]

Default Constructor.

6.846.1.2 decaf::net::UnknownHostException::UnknownHostException ( const Exception & ex )  
throw () [inline]

Conversion Constructor from some other Exception.

### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.846.1.3 decaf::net::UnknownHostException::UnknownHostException ( const  
**UnknownHostException** & ex ) throw () [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.846.1.4 `decaf::net::UnknownHostException::UnknownHostException ( const char * file,  
const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.846.1.5 `decaf::net::UnknownHostException::UnknownHostException ( const std::exception *  
cause ) throw ()` [inline]

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.846.1.6 `decaf::net::UnknownHostException::UnknownHostException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.846.1.7 virtual decaf::net::UnknownHostException::~~UnknownHostException ( ) throw ()  
[inline, virtual]

## 6.846.2 Member Function Documentation

6.846.2.1 virtual UnknownHostException\* decaf::net::UnknownHostException::clone ( )  
const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownHostException.h**

## 6.847 decaf::net::UnknownServiceException Class Reference

```
#include <src/main/decaf/net/UnknownServiceException.h>
```

Inheritance diagram for decaf::net::UnknownServiceException:

### Public Member Functions

- **UnknownServiceException** () throw ()  
*Default Constructor.*
- **UnknownServiceException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **UnknownServiceException** (const **UnknownServiceException** &ex) throw ()  
*Copy Constructor.*
- **UnknownServiceException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnknownServiceException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnknownServiceException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **UnknownServiceException** \* **clone** () const  
*Clones this exception.*
- virtual ~**UnknownServiceException** () throw ()

### 6.847.1 Constructor & Destructor Documentation

6.847.1.1 `decaf::net::UnknownServiceException::UnknownServiceException ( ) throw ()`  
[inline]

Default Constructor.

6.847.1.2 `decaf::net::UnknownServiceException::UnknownServiceException ( const Exception & ex ) throw ()` [inline]

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.847.1.3 `decaf::net::UnknownServiceException::UnknownServiceException ( const UnknownServiceException & ex ) throw ()` [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.847.1.4 `decaf::net::UnknownServiceException::UnknownServiceException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message



6.847.1.5 `decaf::net::UnknownServiceException::UnknownServiceException ( const std::exception * cause ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.847.1.6 `decaf::net::UnknownServiceException::UnknownServiceException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.847.1.7 `virtual decaf::net::UnknownServiceException::~~UnknownServiceException ( ) throw ()` `[inline, virtual]`

## 6.847.2 Member Function Documentation

6.847.2.1 `virtual UnknownServiceException* decaf::net::UnknownServiceException::clone ( )const` `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownServiceException.h`

## 6.848 decaf::io::UnsupportedEncodingException Class Reference

Thrown when the the Character Encoding is not supported.

```
#include <src/main/decaf/io/UnsupportedEncodingException.h>
```

Inheritance diagram for decaf::io::UnsupportedEncodingException:

### Public Member Functions

- **UnsupportedEncodingException** () throw ()  
*Default Constructor.*
- **UnsupportedEncodingException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **UnsupportedEncodingException** (const **UnsupportedEncodingException** &ex) throw ()  
*Copy Constructor.*
- **UnsupportedEncodingException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UnsupportedEncodingException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UnsupportedEncodingException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **UnsupportedEncodingException** \* clone () const  
*Clones this exception.*
- virtual ~**UnsupportedEncodingException** () throw ()

### 6.848.1 Detailed Description

Thrown when the the Character Encoding is not supported.

**Since**

1.0

### 6.848.2 Constructor & Destructor Documentation

6.848.2.1 decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( )  
throw () [inline]

Default Constructor.

6.848.2.2 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const lang::Exception & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.848.2.3 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const UnsupportedEncodingException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.848.2.4 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.848.2.5 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const std::exception * cause ) throw () [inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.848.2.6 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException ( const char * file, const int lineNumber, const char * msg, ... ) throw () [inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.848.2.7 `virtual decaf::io::UnsupportedEncodingException::~~UnsupportedEncodingException ( ) throw () [inline, virtual]`

### 6.848.3 Member Function Documentation

6.848.3.1 `virtual UnsupportedEncodingException* decaf::io::UnsupportedEncodingException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UnsupportedEncodingException.h`

## 6.849 decaf::lang::exceptions::UnsupportedOperationException Class Reference

```
#include <src/main/decaf/lang/exceptions/UnsupportedOperationException.h>
```

Inheritance diagram for `decaf::lang::exceptions::UnsupportedOperationException`:

#### Public Member Functions

- **UnsupportedOperationException** () throw ()

*Default Constructor.*

- **UnsupportedOperationException** (const **Exception** &ex) throw ()

*Conversion Constructor from some other **Exception** (p. 1794).*

- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()

*Copy Constructor.*

- **UnsupportedOperationException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- **UnsupportedOperationException** (const std::exception \*cause) throw ()

*Constructor.*

- **UnsupportedOperationException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()

*Constructor - Initializes the file name and line number where this message occurred.*

- virtual **UnsupportedOperationException** \* clone () const

*Clones this exception.*

- virtual ~**UnsupportedOperationException** () throw ()

## 6.849.1 Constructor & Destructor Documentation

6.849.1.1 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException  
( ) throw () [inline]

Default Constructor.

6.849.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException  
( const **Exception** & ex ) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1794).

### Parameters

ex	An exception that should become this type of <b>Exception</b> (p. 1794)
----	---

6.849.1.3 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException  
( const **UnsupportedOperationException** & ex ) throw () [inline]

Copy Constructor.

### Parameters

ex	An exception that should become this type of <b>Exception</b> (p. 1794)
----	---

6.849.1.4 `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException`  
`( const char * file, const int lineNumber, const std::exception * cause, const char *  
msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.849.1.5 `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException`  
`( const std::exception * cause ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>cause</i>	<b>Pointer</b> (p.2896) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.849.1.6 `decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException`  
`( const char * file, const int lineNumber, const char * msg, ... ) throw ()`  
`[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

6.849.1.7 `virtual decaf::lang::exceptions::UnsupportedOperationException::~~UnsupportedOperationException`  
`( ) throw ()` `[inline, virtual]`

## 6.849.2 Member Function Documentation

```
6.849.2.1 virtual UnsupportedOperationException*
    decaf::lang::exceptions::UnsupportedOperationException::clone ( ) const
    [inline, virtual]
```

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

an new **Exception** (p. 1794) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1797).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 3117).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**UnsupportedOperationException.h**

## 6.850 cms::UnsupportedOperationException Class Reference

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

```
#include <src/main/cms/UnsupportedOperationException.h>
```

Inheritance diagram for cms::UnsupportedOperationException:

### Public Member Functions

- **UnsupportedOperationException** () throw ()
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception \*cause) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception \*cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**UnsupportedOperationException** () throw ()

### 6.850.1 Detailed Description

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

### Since

2.0

## 6.850.2 Constructor & Destructor Documentation

- 6.850.2.1 `cms::UnsupportedOperationException::UnsupportedOperationException ( ) throw ()`
- 6.850.2.2 `cms::UnsupportedOperationException::UnsupportedOperationException ( const UnsupportedOperationException & ex ) throw ()`
- 6.850.2.3 `cms::UnsupportedOperationException::UnsupportedOperationException ( const std::string & message, const std::exception * cause ) throw ()`
- 6.850.2.4 `cms::UnsupportedOperationException::UnsupportedOperationException ( const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace ) throw ()`
- 6.850.2.5 `virtual cms::UnsupportedOperationException::~~UnsupportedOperationException ( ) throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/UnsupportedOperationException.h`

## 6.851 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 3853) as defined by RFC 2396.

```
#include <src/main/decaf/net/URI.h>
```

Inheritance diagram for `decaf::net::URI`:

### Public Member Functions

- **URI** ()  
*Default Constructor, same as calling a Constructor with all fields empty.*
- **URI** (const **URI** &uri) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3853) as a copy of another **URI** (p. 3853).*
- **URI** (const std::string &uri) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3853) from the given string.*
- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3853) from the given components.*
- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3853) from the given components.*



- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3853) from the given components.*
- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment) throw ( URISyntaxException )  
*Constructs a **URI** (p. 3853) from the given components.*
- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const  
*Compares this object with the specified object for order.*
- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **URI** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- std::string **getAuthority** () const
- std::string **getFragment** () const
- std::string **getHost** () const
- std::string **getPath** () const
- int **getPort** () const
- std::string **getQuery** () const
- std::string **getScheme** () const
- std::string **getUserInfo** () const
- std::string **getRawAuthority** () const  
*Returns the raw authority component of this **URI** (p. 3853).*
- std::string **getRawFragment** () const  
*Returns the raw fragment component of this **URI** (p. 3853).*
- std::string **getRawPath** () const  
*Returns the raw path component of this **URI** (p. 3853).*
- std::string **getRawQuery** () const  
*Returns the raw query component of this **URI** (p. 3853).*
- std::string **getRawSchemeSpecificPart** () const  
*Returns the raw scheme-specific part of this **URI** (p. 3853).*
- std::string **getSchemeSpecificPart** () const  
*Returns the decoded scheme-specific part of this **URI** (p. 3853).*
- std::string **getRawUserInfo** () const  
*Returns the raw user-information component of this **URI** (p. 3853).*
- bool **isAbsolute** () const  
*Tells whether or not this **URI** (p. 3853) is absolute.*
- bool **isOpaque** () const  
*Tells whether or not this **URI** (p. 3853) is opaque.*
- **URI normalize** () const  
*Normalizes this **URI**'s path.*

- **URI parseServerAuthority** () const throw ( URISyntaxException )  
*Attempts to parse this URI's authority component, if defined, into user-information, host, and port components.*
- **URI relativize** (const **URI** &uri) const  
*Relativizes the given **URI** (p. 3853) against this **URI** (p. 3853).*
- **URI resolve** (const std::string &str) const throw ( lang::exceptions::IllegalArgumentException )  
*Constructs a new **URI** (p. 3853) by parsing the given string and then resolving it against this **URI** (p. 3853).*
- **URI resolve** (const **URI** &uri) const  
*Resolves the given **URI** (p. 3853) against this **URI** (p. 3853).*
- std::string **toString** () const  
*Returns the content of this **URI** (p. 3853) as a string.*
- **URL toURL** () const throw ( MalformedURLException, lang::exceptions::IllegalArgumentException )  
*Constructs a **URL** (p. 3891) from this **URI** (p. 3853).*

### Static Public Member Functions

- static **URI create** (const std::string uri) throw ( lang::exceptions::IllegalArgumentException )  
*Creates a **URI** (p. 3853) by parsing the given string.*

#### 6.851.1 Detailed Description

This class represents an instance of a **URI** (p. 3853) as defined by RFC 2396.

#### 6.851.2 Constructor & Destructor Documentation

##### 6.851.2.1 decaf::net::URI::URI ( )

Default Constructor, same as calling a Constructor with all fields empty.

##### 6.851.2.2 decaf::net::URI::URI ( const **URI** & uri ) throw ( URISyntaxException )

Constructs a **URI** (p. 3853) as a copy of another **URI** (p. 3853).

#### Parameters

<i>uri</i>	- uri to copy
------------	---------------

**6.851.2.3 decaf::net::URI::URI ( const std::string & *uri* ) throw ( URISyntaxException )**

Constructs a **URI** (p. 3853) from the given string.

**Parameters**

<i>uri</i>	- string uri to parse.
------------	------------------------

**6.851.2.4 decaf::net::URI::URI ( const std::string & *scheme*, const std::string & *ssp*, const std::string & *fragment* ) throw ( URISyntaxException )**

Constructs a **URI** (p. 3853) from the given components.

**Parameters**

<i>scheme</i>	- the uri scheme
<i>ssp</i>	- Scheme specific part
<i>fragment</i>	- Fragment

**6.851.2.5 decaf::net::URI::URI ( const std::string & *scheme*, const std::string & *userInfo*, const std::string & *host*, int *port*, const std::string & *path*, const std::string & *query*, const std::string & *fragment* ) throw ( URISyntaxException )**

Constructs a **URI** (p. 3853) from the given components.

**Parameters**

<i>scheme</i>	- Scheme name
<i>userInfo</i>	- User name and authorization information
<i>host</i>	- Host name
<i>port</i>	- Port number
<i>path</i>	- Path
<i>query</i>	- Query
<i>fragment</i>	- Fragment

**6.851.2.6 decaf::net::URI::URI ( const std::string & *scheme*, const std::string & *host*, const std::string & *path*, const std::string & *fragment* ) throw ( URISyntaxException )**

Constructs a **URI** (p. 3853) from the given components.

**Parameters**

<i>scheme</i>	- Scheme name
<i>host</i>	- Host name
<i>path</i>	- Path
<i>fragment</i>	- Fragment

6.851.2.7 `decaf::net::URI::URI ( const std::string & scheme, const std::string & authority, const std::string & path, const std::string & query, const std::string & fragment ) throw ( URISyntaxException )`

Constructs a **URI** (p. 3853) from the given components.

#### Parameters

<i>scheme</i>	- Scheme name
<i>authority</i>	- Authority
<i>path</i>	- Path
<i>query</i>	- Query
<i>fragment</i>	- Fragment

6.851.2.8 `virtual decaf::net::URI::~~URI ( ) [inline, virtual]`

### 6.851.3 Member Function Documentation

6.851.3.1 `virtual int decaf::net::URI::compareTo ( const URI & value ) const [virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

#### Parameters

<i>value</i>	- the value to compare to this one.
--------------	-------------------------------------

#### Returns

zero if equal minus one if less than and one if greater than.

6.851.3.2 `static URI decaf::net::URI::create ( const std::string uri ) throw ( lang::exceptions::IllegalArgumentException ) [static]`

Creates a **URI** (p. 3853) by parsing the given string.

This convenience factory method works as if by invoking the `URI(string)` constructor; any **URISyntaxException** (p. 3880) thrown by the constructor is caught and wrapped in a new `IllegalArgumentException` object, which is then thrown.

#### Parameters

<i>uri</i>	- <b>URI</b> (p. 3853) string to parse
------------	--

#### Exceptions

<i>IllegalArgumentException</i>	
---------------------------------	--

6.851.3.3 virtual bool decaf::net::URI::equals ( const URI & *value* ) const [virtual]

**Returns**

true if this value is considered equal to the passed value.

6.851.3.4 std::string decaf::net::URI::getAuthority ( ) const

**Returns**

the decoded authority component of this **URI** (p. 3853).

6.851.3.5 std::string decaf::net::URI::getFragment ( ) const

**Returns**

the decoded fragment component of this **URI** (p. 3853).

6.851.3.6 std::string decaf::net::URI::getHost ( ) const

**Returns**

the host component of this **URI** (p. 3853).

6.851.3.7 std::string decaf::net::URI::getPath ( ) const

**Returns**

the path component of this **URI** (p. 3853).

6.851.3.8 int decaf::net::URI::getPort ( ) const

**Returns**

the port component of this **URI** (p. 3853).

6.851.3.9 std::string decaf::net::URI::getQuery ( ) const

**Returns**

the query component of this **URI** (p. 3853).

#### 6.851.3.10 `std::string decaf::net::URI::getRawAuthority ( ) const`

Returns the raw authority component of this **URI** (p. 3853).

The authority component of a **URI** (p. 3853), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

##### Returns

the raw authority component of the **URI** (p. 3853)

#### 6.851.3.11 `std::string decaf::net::URI::getRawFragment ( ) const`

Returns the raw fragment component of this **URI** (p. 3853).

The fragment component of a **URI** (p. 3853), if defined, only contains legal **URI** (p. 3853) characters.

##### Returns

the raw fragment component of this **URI** (p. 3853)

#### 6.851.3.12 `std::string decaf::net::URI::getRawPath ( ) const`

Returns the raw path component of this **URI** (p. 3853).

The path component of a **URI** (p. 3853), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

##### Returns

the raw path component of this **URI** (p. 3853)

#### 6.851.3.13 `std::string decaf::net::URI::getRawQuery ( ) const`

Returns the raw query component of this **URI** (p. 3853).

The query component of a **URI** (p. 3853), if defined, only contains legal **URI** (p. 3853) characters.

##### Returns

the raw query component of the **URI** (p. 3853).

#### 6.851.3.14 `std::string decaf::net::URI::getRawSchemeSpecificPart ( ) const`

Returns the raw scheme-specific part of this **URI** (p. 3853).

The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 3853) only contains legal **URI** (p. 3853) characters.

##### Returns

the raw scheme special part of the uri

#### 6.851.3.15 `std::string decaf::net::URI::getRawUserInfo ( ) const`

Returns the raw user-information component of this **URI** (p. 3853).

The user-information component of a **URI** (p. 3853), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

##### Returns

the raw user-information component of the **URI** (p. 3853)

#### 6.851.3.16 `std::string decaf::net::URI::getScheme ( ) const`

##### Returns

the scheme component of this **URI** (p. 3853)

#### 6.851.3.17 `std::string decaf::net::URI::getSchemeSpecificPart ( ) const`

Returns the decoded scheme-specific part of this **URI** (p. 3853).

The string returned by this method is equal to that returned by the `getRawSchemeSpecificPart` method except that all sequences of escaped octets are decoded.

##### Returns

the raw scheme specific part of the uri.

#### 6.851.3.18 `std::string decaf::net::URI::getUserInfo ( ) const`

##### Returns

the user info component of this **URI** (p. 3853)

**6.851.3.19** `bool decaf::net::URI::isAbsolute ( ) const`

Tells whether or not this **URI** (p. 3853) is absolute.

A **URI** (p. 3853) is absolute if, and only if, it has a scheme component.

**Returns**

true if, and only if, this **URI** (p. 3853) is absolute

**6.851.3.20** `bool decaf::net::URI::isOpaque ( ) const`

Tells whether or not this **URI** (p. 3853) is opaque.

A **URI** (p. 3853) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 3853) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

**Returns**

true if, and only if, this **URI** (p. 3853) is opaque

**6.851.3.21** `URI decaf::net::URI::normalize ( ) const`

Normalizes this URI's path.

If this **URI** (p. 3853) is opaque, or if its path is already in normal form, then this **URI** (p. 3853) is returned. Otherwise a new **URI** (p. 3853) is constructed that is identical to this **URI** (p. 3853) except that its path is computed by normalizing this URI's path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed. 2. If a ".." segment is preceded by a non-".." segment then both of these segments are removed. This step is repeated until it is no longer applicable. 3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 3853) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 3853) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more ".." segments if there were insufficient non-".." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

**Returns**

A **URI** (p. 3853) equivalent to this **URI** (p. 3853), but whose path is in normal form

**6.851.3.22** `virtual bool decaf::net::URI::operator< ( const URI & value ) const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.



This

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

6.851.3.23 `virtual bool decaf::net::URI::operator==( const URI & value ) const` [virtual]

Compares equality between this object and the one passed.

#### Parameters

<i>value</i>	- the value to be compared to this one.
--------------	---

#### Returns

true if this object is equal to the one passed.

6.851.3.24 `URI decaf::net::URI::parseServerAuthority ( ) const throw ( URISyntaxException )`

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components.

If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p. 3853) has no authority component, this method simply returns this **URI** (p. 3853).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

#### Returns

A **URI** (p. 3853) whose authority field has been parsed as a server-based authority

#### Exceptions

<b><i>URISyntaxException</i></b> (p. 3880)	- If the authority component of this <b>URI</b> (p. 3853) is defined but cannot be parsed as a server-based authority.
---	--

**6.851.3.25 URI decaf::net::URI::relativize ( const URI & uri ) const**

Relativizes the given **URI** (p. 3853) against this **URI** (p. 3853).

The relativization of the given **URI** (p. 3853) against this **URI** (p. 3853) is computed as follows:

1. If either this **URI** (p. 3853) or the given **URI** (p. 3853) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 3853) is not a prefix of the path of the given **URI** (p. 3853), then the given **URI** (p. 3853) is returned.
2. Otherwise a new relative hierarchical **URI** (p. 3853) is constructed with query and fragment components taken from the given **URI** (p. 3853) and with a path component computed by removing this URI's path from the beginning of the given URI's path.

**Parameters**

<i>uri</i>	- The <b>URI</b> (p. 3853) to be relativized against this <b>URI</b> (p. 3853)
------------	--

**Returns**

The resulting **URI** (p. 3853)

**6.851.3.26 URI decaf::net::URI::resolve ( const std::string & str ) const throw ( lang::exceptions::IllegalArgumentException )**

Constructs a new **URI** (p. 3853) by parsing the given string and then resolving it against this **URI** (p. 3853).

This convenience method works as if invoking it were equivalent to evaluating the expression `resolve( URI::create( str ) )`.

**Parameters**

<i>str</i>	- The string to be parsed into a <b>URI</b> (p. 3853)
------------	---

**Returns**

The resulting **URI** (p. 3853)

**Exceptions**

<i>IllegalArgumentException</i>	- If the given string violates RFC 2396
---------------------------------	---

**6.851.3.27 URI decaf::net::URI::resolve ( const URI & uri ) const**

Resolves the given **URI** (p. 3853) against this **URI** (p. 3853).

If the given **URI** (p. 3853) is already absolute, or if this **URI** (p. 3853) is opaque, then a copy of the given **URI** (p. 3853) is returned.

If the given URI's fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 3853) with the given fragment but with all other components equal to those of this **URI** (p. 3853) is returned. This allows a **URI** (p. 3853) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 3853).

Otherwise this method constructs a new hierarchical **URI** (p. 3853) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 3853) is constructed with this URI's scheme and the given URI's query and fragment components. 2. If the given **URI** (p. 3853) has an authority component then the new URI's authority and path are taken from the given **URI** (p. 3853). 3. Otherwise the new URI's authority component is copied from this **URI** (p. 3853), and its path is computed as follows:

1. If the given URI's path is absolute then the new URI's path is taken from the given **URI** (p. 3853). 2. Otherwise the given URI's path is relative, and so the new URI's path is computed by resolving the path of the given **URI** (p. 3853) against the path of this **URI** (p. 3853). This is done by concatenating all but the last segment of this URI's path, if any, with the given URI's path and then normalizing the result as if by invoking the `normalize` method.

The result of this method is absolute if, and only if, either this **URI** (p. 3853) is absolute or the given **URI** (p. 3853) is absolute.

#### Parameters

<i>uri</i>	- The <b>URI</b> (p. 3853) to be resolved against this <b>URI</b> (p. 3853)
------------	---

#### Returns

The resulting **URI** (p. 3853)

**6.851.3.28** `std::string decaf::net::URI::toString ( ) const`

Returns the content of this **URI** (p. 3853) as a string.

If this **URI** (p. 3853) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 3853) was created by normalization, resolution, or relativization, and so a string is constructed from this URI's components according to the rules specified in RFC 2396, section 5.2, step 7.

#### Returns

the string form of this **URI** (p. 3853)

**6.851.3.29** `URL decaf::net::URI::toURL ( ) const throw ( MalformedURLException, lang::exceptions::IllegalArgumentException )`

Constructs a **URL** (p. 3891) from this **URI** (p. 3853).

This convenience method works as if invoking it were equivalent to evaluating the expression `new URL (p. 3891)(this.toString())` after first checking that this **URI** (p. 3853) is absolute.

### Returns

A **URL** (p. 3891) constructed from this **URI** (p. 3853)

### Exceptions

<i><b>IllegalArgumentEx- ception</b></i>	- If this <b>URL</b> (p. 3891) is not absolute
<i><b>MalformedURLEx- ception</b></i> (p. 2416)	- If a protocol handler for the <b>URL</b> (p. 3891) could not be found, or if some other error occurred while constructing the <b>URL</b> (p. 3891)

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URI.h`

## 6.852 decaf::internal::net::URLEncoderDecoder Class Reference

```
#include <src/main/decaf/internal/net/URLEncoderDecoder.h>
```

### Public Member Functions

- **URLEncoderDecoder** ()
- virtual **~URLEncoderDecoder** ()

### Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal) throw ( decaf::net::URISyntaxException )  
*Validate a string by checking if it contains any characters other than:*
- static void **validateSimple** (const std::string &s, const std::string &legal) throw ( decaf::net::URISyntaxException )  
*Validate a string by checking if it contains any characters other than:*
- static std::string **quotellegal** (const std::string &s, const std::string &legal)  
*All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by '%'*
- static std::string **encodeOthers** (const std::string &s)  
*Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.*
- static std::string **decode** (const std::string &s)  
*Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.*

## 6.852.1 Constructor & Destructor Documentation

6.852.1.1 `decaf::internal::net::URLEncoderDecoder::URLEncoderDecoder ( )`

6.852.1.2 `virtual decaf::internal::net::URLEncoderDecoder::~~URLEncoderDecoder ( )`  
[inline, virtual]

## 6.852.2 Member Function Documentation

6.852.2.1 `static std::string decaf::internal::net::URLEncoderDecoder::decode ( const std::string & s )` [static]

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type using the UTF-8 encoding scheme.

'%' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C \$%"

### Parameters

<code>s</code>	- The encoded string.
----------------	-----------------------

### Returns

The decoded version.

6.852.2.2 `static std::string decaf::internal::net::URLEncoderDecoder::encodeOthers ( const std::string & s )` [static]

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

They are converted into their hexadecimal value prepended by '%'.

For example: Euro currency symbol -> "%E2%82%AC".

### Parameters

<code>s</code>	- the string to be converted
----------------	------------------------------

### Returns

the converted string

6.852.2.3 `static std::string decaf::internal::net::URLEncoderDecoder::quotelllegal ( const std::string & s, const std::string & legal )` [static]

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by '%'.

For example: '#' -> %23

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

#### Parameters

<i>s</i>	- the string to be converted
<i>legal</i>	- the characters allowed to be preserved in the string <i>s</i>

#### Returns

converted string

```
6.852.2.4 static void decaf::internal::net::URIEncoderDecoder::validate ( const std::string
& s, const std::string & legal ) throw ( decaf::net::URISyntaxException )
[static]
```

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter
4. characters that are not ISO Control or are not ISO Space characters)

#### Parameters

<i>s</i>	- the string to be validated
<i>legal</i>	- the characters allowed in the string <i>s</i>

```
6.852.2.5 static void decaf::internal::net::URIEncoderDecoder::validateSimple ( const std::string
& s, const std::string & legal ) throw ( decaf::net::URISyntaxException )
[static]
```

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter

#### Parameters

<i>s</i>	- the string to be validated
<i>legal</i>	- the characters allowed in the string <i>s</i>

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIEncoderDecoder.h**

## 6.853 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

```
#include <src/main/decaf/internal/net/URIHelper.h>
```

## Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)  
*Setup the **URIHelper** (p. 3867) with values assigned to the various fields that are used in the validation process.*
- **URIHelper** ()  
*Sets up the filter strings with sane defaults.*
- virtual ~**URIHelper** ()
- **URIType parseURI** (const std::string &uri, bool forceServer) throw ( decaf::net::URISyntaxException )  
*Parse the passed in URI.*
- void **validateScheme** (const std::string &uri, const std::string &scheme, int index) throw ( decaf::net::URISyntaxException )  
*Validate the schema portin of the URI.*
- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Ssp Segment contains no invalid encodings.*
- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Authority Segment contains no invalid encodings.*
- void **validatePath** (const std::string &uri, const std::string &path, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Path Segment contains no invalid encodings.*
- void **validateQuery** (const std::string &uri, const std::string &query, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI Query Segment contains no invalid encodings.*
- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Validate that the URI fragment contains no invalid encodings.*
- **URIType parseAuthority** (bool forceServer, const std::string &authority) throw ( decaf::net::URISyntaxException )  
*determine the host, port and user-info if the authority parses successfully to a server based authority*
- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size\_t index) throw ( decaf::net::URISyntaxException )  
*Check the supplied user info for validity.*
- bool **isValidHost** (bool forceServer, const std::string &host) throw ( decaf::net::URISyntaxException )  
*distinguish between IPv4, IPv6, domain name and validate it based on its type*
- bool **isValidDomainName** (const std::string &host)  
*Validates the string past to determine if it is a well formed domain name.*
- bool **isValidIPv4Address** (const std::string &host)

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.

- bool **isValidIP6Address** (const std::string &ipAddress)  
Determines if the given address is valid according to the IPv6 spec.
- bool **isValidIP4Word** (const std::string &word)  
Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.
- bool **isValidHexChar** (char c)  
Determines if the given char is a valid Hex char.

### 6.853.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

### 6.853.2 Constructor & Destructor Documentation

6.853.2.1 `decaf::internal::net::URIHelper::URIHelper ( const std::string & unreserved, const std::string & punct, const std::string & reserved, const std::string & someLegal, const std::string & allLegal )`

Setup the **URIHelper** (p. 3867) with values assigned to the various fields that are used in the validation process.

The defaults are overridden by these values.

#### Parameters

<i>unreserved</i>	- characters not reserved for use.
<i>punct</i>	- allowable punctuation symbols.
<i>reserved</i>	- characters not allowed for general use in the URI.
<i>someLegal</i>	- characters that are legal in certain cases.
<i>allLegal</i>	- characters that are always legal.

6.853.2.2 `decaf::internal::net::URIHelper::URIHelper ( )`

Sets up the filter strings with sane defaults.

6.853.2.3 `virtual decaf::internal::net::URIHelper::~URIHelper ( ) [inline, virtual]`

### 6.853.3 Member Function Documentation

6.853.3.1 `bool decaf::internal::net::URIHelper::isValidDomainName ( const std::string & host )`

Validates the string past to determine if it is a well formed domain name.



**Parameters**

<i>host</i>	- domain name to validate.
-------------	----------------------------

**Returns**

true if host is well formed.

**6.853.3.2 bool decaf::internal::net::URIHelper::isValidHexChar ( char *c* )**

Determines if the given char is a valid Hex char.

Valid chars are A-F (upper or lower case) and 0-9.

**Parameters**

<i>c</i>	- char to inspect
----------	-------------------

**Returns**

true if *c* is a valid hex char.

**6.853.3.3 bool decaf::internal::net::URIHelper::isValidHost ( bool *forceServer*, const std::string & *host* ) throw ( decaf::net::URISyntaxException )**

distinguish between IPv4, IPv6, domain name and validate it based on its type

**Parameters**

<i>forceServer</i>	- true if the forceServer mode should be active.
<i>host</i>	- Host string to validate.

**Returns**

true if the host value if a valid domain name.

**Exceptions**

<i>URISyntaxException</i>	if the host is invalid and forceServer is true.
---------------------------	---

**6.853.3.4 bool decaf::internal::net::URIHelper::isValidIP4Word ( const std::string & *word* )**

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

**Parameters**

<i>word</i>	- string value to check.
-------------	--------------------------

**Returns**

true if the word is a valid IPv4 word.

**6.853.3.5** `bool decaf::internal::net::URIHelper::isValidIP6Address ( const std::string & ipAddress )`

Determines if the given address is valid according to the IPv6 spec.

**Parameters**

<i>ipAddress</i>	- string ip address value to validate.
------------------	--

**Returns**

true if the address string is valid.

**6.853.3.6** `bool decaf::internal::net::URIHelper::isValidIPv4Address ( const std::string & host )`

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.

and XXX is not greater than 255.

**Parameters**

<i>host</i>	- IPv4 address string to parse.
-------------	---------------------------------

**Returns**

true if host is a well formed IPv4 address.

**6.853.3.7** `URIType decaf::internal::net::URIHelper::parseAuthority ( bool forceServer, const std::string & authority ) throw ( decaf::net::URISyntaxException )`

determine the host, port and user-info if the authority parses successfully to a server based authority

behavior in error cases: if forceServer is true, throw URISyntaxException with the proper diagnostic messages. if forceServer is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where URISyntaxException is thrown regardless of the forceServer parameter e.g. mal-formed ipv6 address

**Parameters**

<i>forceServer</i>	
<i>authority</i>	

**Returns**

a **URIType** (p. 3884) instance containing the parsed data.

**Exceptions**

<i>URISyntaxException</i>	
---------------------------	--

**6.853.3.8** **URIType** decaf::internal::net::URIHelper::parseURI ( const std::string & *uri*, bool *forceServer* ) throw ( decaf::net::URISyntaxException )

Parse the passed in URI.

**Parameters**

<i>uri</i>	- the URI to Parse
<i>forceServer</i>	- if true invalid URI data throws an Exception

**Returns**

a **URIType** (p. 3884) instance containing the parsed data.

**Exceptions**

<i>URISyntaxException</i>	if forceServer is true and the URI is invalid.
---------------------------	--

**6.853.3.9** void decaf::internal::net::URIHelper::validateAuthority ( const std::string & *uri*, const std::string & *authority*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Authority Segment contains no invalid encodings.

**Parameters**

<i>uri</i>	- the full uri.
<i>authority</i>	- the Authority to check.
<i>index</i>	- position in the uri where Authority starts.

**Exceptions**

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

**6.853.3.10** void decaf::internal::net::URIHelper::validateFragment ( const std::string & *uri*, const std::string & *fragment*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI fragment contains no invalid encodings.

**Parameters**

<i>uri</i>	- the full uri.
<i>fragment</i>	- the fragment to check.
<i>index</i>	- position in the uri where fragment starts.

**Exceptions**

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.853.3.11 void decaf::internal::net::URIHelper::validatePath ( const std::string & *uri*, const std::string & *path*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Path Segment contains no invalid encodings.

**Parameters**

<i>uri</i>	- the full uri.
<i>path</i>	- the path to check.
<i>index</i>	- position in the uri where path starts.

**Exceptions**

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.853.3.12 void decaf::internal::net::URIHelper::validateQuery ( const std::string & *uri*, const std::string & *query*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Query Segment contains no invalid encodings.

**Parameters**

<i>uri</i>	- the full uri.
<i>query</i>	- the query to check.
<i>index</i>	- position in the uri where fragment starts.

**Exceptions**

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.853.3.13 void decaf::internal::net::URIHelper::validateScheme ( const std::string & *uri*, const std::string & *scheme*, int *index* ) throw ( decaf::net::URISyntaxException )

Validate the schema portin of the URI.

**Parameters**

<i>uri</i>	- the URI to check.
<i>scheme</i>	- the schema section of the URI.
<i>index</i>	- index in uri where schema starts.

### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.853.3.14 void decaf::internal::net::URIHelper::validateSsp ( const std::string & *uri*, const std::string & *ssp*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Validate that the URI Ssp Segment contains no invalid encodings.

### Parameters

<i>uri</i>	- the full uri.
<i>ssp</i>	- the SSP to check.
<i>index</i>	- position in the uri where Ssp starts.

### Exceptions

<i>URISyntaxException</i>	if the fragment has errors.
---------------------------	-----------------------------

6.853.3.15 void decaf::internal::net::URIHelper::validateUserinfo ( const std::string & *uri*, const std::string & *userinfo*, std::size\_t *index* ) throw ( decaf::net::URISyntaxException )

Check the supplied user info for validity.

### Parameters

<i>uri</i>	- the uri to parse.
<i>userinfo</i>	- supplied user info
<i>index</i>	- index into the URI string where the data is located.

### Returns

true if valid

### Exceptions

<i>URISyntaxException</i>	if an error occurs
---------------------------	--------------------

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIHelper.h**

## 6.854 activemq::transport::failover::URIPool Class Reference

```
#include <src/main/activemq/transport/failover/URIPool.h>
```

### Public Member Functions

- **URIPool** ()  
*Create an Empty URI Pool.*
- **URIPool** (const **decaf::util::List**< **URI** > &uris)  
*Creates a new URI Pool using the given list as the initial Free List.*
- **~URIPool** ()
- **URI getURI** () throw ( **decaf::lang::exceptions::NoSuchElementException** )  
*Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.*
- void **addURI** (const **URI** &uri)  
*Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.*
- void **addURIs** (const **StlList**< **URI** > &uris)  
*Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.*
- void **removeURI** (const **URI** &uri)  
*Remove a given URI from the Free List.*
- bool **isRandomize** () const  
*Is the URI that is given randomly picked from the pool or is each one taken in sequence.*
- void **setRandomize** (bool value)  
*Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.*

### 6.854.1 Constructor & Destructor Documentation

#### 6.854.1.1 activemq::transport::failover::URIPool::URIPool ( )

Create an Empty URI Pool.

#### 6.854.1.2 activemq::transport::failover::URIPool::URIPool ( const **decaf::util::List**< **URI** > &uris )

Creates a new URI Pool using the given list as the initial Free List.

#### Parameters

<i>uris</i>	- List of URI to place in the Pool.
-------------	-------------------------------------

6.854.1.3 `activemq::transport::failover::URIPool::~~URIPool ( )`

## 6.854.2 Member Function Documentation

6.854.2.1 `void activemq::transport::failover::URIPool::addURI ( const URI & uri )`

Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.

### Parameters

<i>uri</i>	- a URI previously taken from the pool.
------------	---

6.854.2.2 `void activemq::transport::failover::URIPool::addURIs ( const StlList< URI > & uris )`

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

### Parameters

<i>uris</i>	- List of URIs to add into the Pool.
-------------	--------------------------------------

6.854.2.3 `URI activemq::transport::failover::URIPool::getURI ( ) throw ( decaf::lang::exceptions::NoSuchElementException )`

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a `NoSuchElementException`.

Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

### Returns

the next free URI in the Pool.

### Exceptions

<i>NoSuchElementException</i>	if there are none free currently.
-------------------------------	-----------------------------------

6.854.2.4 `bool activemq::transport::failover::URIPool::isRandomize ( ) const [inline]`

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

**Returns**

true if URI gets are random.

**6.854.2.5** `void activemq::transport::failover::URIPool::removeURI ( const URI & uri )`

Remove a given URI from the Free List.

**Parameters**

<code>uri</code>	- the URI to find and remove from the free list
------------------	---

**6.854.2.6** `void activemq::transport::failover::URIPool::setRandomize ( bool value )`  
`[inline]`

Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

**Parameters**

<code>value</code>	- true indicates URI gets are random.
--------------------	---------------------------------------

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/URIPool.h`

**6.855 activemq::util::URISupport Class Reference**

```
#include <src/main/activemq/util/URISupport.h>
```

**Static Public Member Functions**

- static void **parseURL** (const std::string &URI, **decaf::util::Properties** &properties) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.*
- static **CompositeData** **parseComposite** (const URI &uri) throw ( decaf::net::URISyntaxException )  
*Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.*
- static **decaf::util::Properties** **parseQuery** (std::string query) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.*



- static void **parseQuery** (std::string query, **decaf::util::Properties** \*properties) throw ( decaf::lang::exceptions::IllegalArgumentException )  
*Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.*
- static std::string **createQueryString** (const **Properties** &options) throw ( decaf::net::URISyntaxException )  
*Given a properties object create a string that can be appended to a URI as a valid Query string.*

### 6.855.1 Member Function Documentation

6.855.1.1 static std::string activemq::util::URISupport::createQueryString ( const **Properties** &options ) throw ( decaf::net::URISyntaxException ) [static]

Given a properties object create a string that can be appended to a URI as a valid Query string.

#### Parameters

<i>options</i>	Properties object containing key / value query values.
----------------	--

#### Returns

a valid URI query string.

#### Exceptions

<i>URISyntaxException</i>	if the string in the Properties object can't be encoded into a valid URI Query string.
---------------------------	--

6.855.1.2 static **CompositeData** activemq::util::URISupport::parseComposite ( const **URI** &uri ) throw ( decaf::net::URISyntaxException ) [static]

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

#### Parameters

<i>uri</i>	- The Composite URI to parse.
------------	-------------------------------

#### Returns

a new **CompositeData** (p. 1191) object with the parsed data

#### Exceptions

<i>URISyntaxException</i>	if the URI is not well formed.
---------------------------	--------------------------------

6.855.1.3 `static void activemq::util::URISupport::parseQuery ( std::string  
query, decaf::util::Properties * properties ) throw (  
decaf::lang::exceptions::IllegalArgumentException ) [static]`

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

#### Parameters

<i>query</i>	- the query string to parse.
<i>properties</i>	- object pointer to get the parsed output.

#### Exceptions

<i>IllegalArgumentException</i>	if the Query string is not well formed.
---------------------------------	---

6.855.1.4 `static decaf::util::Properties activemq::util::URISupport::parseQuery ( std::string  
query ) throw ( decaf::lang::exceptions::IllegalArgumentException )  
[static]`

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

#### Parameters

<i>query</i>	The query string to parse and extract the encoded properties.
--------------	---

#### Returns

Properties object with the parsed output.

#### Exceptions

<i>IllegalArgumentException</i>	if the Query string is not well formed.
---------------------------------	---

6.855.1.5 `static void activemq::util::URISupport::parseURL ( const std::string  
& URI, decaf::util::Properties & properties ) throw (  
decaf::lang::exceptions::IllegalArgumentException ) [static]`

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

#### Parameters

<i>URI</i>	a Broker URI to parse
<i>properties</i>	a Properties object to set the parsed values in

**Exceptions**

<i>IllegalArgumentEx- ception</i>	if the passed URI is invalid
---------------------------------------	------------------------------

The documentation for this class was generated from the following file:

- src/main/activemq/util/**URISupport.h**

**6.856 decaf::net::URISyntaxException Class Reference**

```
#include <src/main/decaf/net/URISyntaxException.h>
```

Inheritance diagram for decaf::net::URISyntaxException:

**Public Member Functions**

- **URISyntaxException** () throw ()  
*Default Constructor.*
- **URISyntaxException** (const Exception &ex) throw ()  
*Conversion Constructor from some other Exception.*
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()  
*Copy Constructor.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::exception \***cause**, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const std::exception \***cause**) throw ()  
*Constructor.*
- **URISyntaxException** (const char \*file, const int lineNumber, const char \*msg DECAF\_UNUSED) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::string &input, const std::string &reason) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **URISyntaxException** (const char \*file, const int lineNumber, const std::string &input, const std::string &reason, int index) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- virtual **URISyntaxException** \* **clone** () const  
*Clones this exception.*
- virtual ~**URISyntaxException** () throw ()
- std::string **getInput** () const
- std::string **getReason** () const
- int **getIndex** () const

### 6.856.1 Constructor & Destructor Documentation

6.856.1.1 `decaf::net::URISyntaxException::URISyntaxException ( ) throw () [inline]`

Default Constructor.

6.856.1.2 `decaf::net::URISyntaxException::URISyntaxException ( const Exception & ex ) throw () [inline]`

Conversion Constructor from some other Exception.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.856.1.3 `decaf::net::URISyntaxException::URISyntaxException ( const URISyntaxException & ex ) throw () [inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	An exception that should become this type of Exception
-----------	--

6.856.1.4 `decaf::net::URISyntaxException::URISyntaxException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.856.1.5 `decaf::net::URISyntaxException::URISyntaxException ( const std::exception * cause ) throw () [inline]`

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.856.1.6 `decaf::net::URISyntaxException::URISyntaxException ( const char * file, const int lineNumber, const char *msg DECAF_UNUSED ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.856.1.7 `decaf::net::URISyntaxException::URISyntaxException ( const char * file, const int lineNumber, const std::string & input, const std::string & reason ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the input string that caused the error and the reason for the error.

**Parameters**

<i>file</i>	The file name where exception occurs.
<i>lineNumber</i>	The line number where the exception occurred.
<i>input</i>	The <b>URL</b> (p. 3891) that caused the exception.
<i>reason</i>	The reason for the failure.

6.856.1.8 `decaf::net::URISyntaxException::URISyntaxException ( const char * file, const int lineNumber, const std::string & input, const std::string & reason, int index ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the input string that caused the error and the reason for the error.

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>input</i>	The input <b>URI</b> (p. 3853) that caused the exception
<i>reason</i>	The reason for the failure.

<i>index</i>	The index in the <b>URI</b> (p. 3853) string where the error occurred.
--------------	--

6.856.1.9 `virtual decaf::net::URISyntaxException::~~URISyntaxException ( ) throw ()`  
`[inline, virtual]`

## 6.856.2 Member Function Documentation

6.856.2.1 `virtual URISyntaxException* decaf::net::URISyntaxException::clone ( ) const`  
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p. 1797).

6.856.2.2 `int decaf::net::URISyntaxException::getIndex ( ) const` `[inline]`

### Returns

the index in the input string where the error occurred or -1

6.856.2.3 `std::string decaf::net::URISyntaxException::getInput ( ) const` `[inline]`

### Returns

the Input string that cause this exception or ""

6.856.2.4 `std::string decaf::net::URISyntaxException::getReason ( ) const` `[inline]`

### Returns

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URISyntaxException.h`

## 6.857 decaf::internal::net::URIType Class Reference

Basic type object that holds data that composes a given URI.

```
#include <src/main/decaf/internal/net/URIType.h>
```

### Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual ~**URIType** ()
- std::string **getSource** () const  
*Gets the source URI string that was parsed to obtain this **URIType** (p. 3884) instance and the resulting data.,*
- void **setSource** (const std::string &source)  
*Sets the source URI string that was parsed to obtain this **URIType** (p. 3884) instance and the resulting data.,*
- std::string **getScheme** () const  
*Gets the Scheme of the URI, e.g.*
- void **setScheme** (const std::string &scheme)  
*Sets the Scheme of the URI, e.g.*
- std::string **getSchemeSpecificPart** () const  
*Gets the Scheme Specific Part of the URI.*
- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)  
*Sets the Scheme Specific Part of the URI.*
- std::string **getAuthority** () const  
*Gets the Authority of the URI.*
- void **setAuthority** (const std::string &authority)  
*Sets the Authority of the URI.*
- std::string **getUserInfo** () const  
*Gets the user info part of the URI, e.g.*
- void **setUserInfo** (const std::string &userinfo)  
*Sets the user info part of the URI, e.g.*
- std::string **getHost** () const  
*Gets the Host name part of the URI.*
- void **setHost** (const std::string &host)  
*Sets the Host name part of the URI.*
- int **getPort** () const  
*Gets the port part of the URI.*
- void **setPort** (int port)  
*Sets the port part of the URI.*
- std::string **getPath** () const  
*Gets the Path part of the URI.*
- void **setPath** (const std::string &path)

- Sets the Path part of the URI.*
- std::string **getQuery** () const  
*Gets the Query part of the URI.*
- void **setQuery** (const std::string &query)  
*Sets the Query part of the URI.*
- std::string **getFragment** () const  
*Gets the Fragment part of the URI.*
- void **setFragment** (const std::string &fragment)  
*Sets the Fragment part of the URI.*
- bool **isOpaque** () const  
*Gets if the URI is Opaque.*
- void **setOpaque** (bool opaque)  
*Sets if the URI is Opaque.*
- bool **isAbsolute** () const  
*Gets if the URI is Absolute.*
- void **setAbsolute** (bool absolute)  
*Sets if the URI is Absolute.*
- bool **isServerAuthority** () const  
*Gets if the URI is a Server Authority.*
- void **setServerAuthority** (bool serverAuthority)  
*Sets if the URI is a Server Authority.*
- bool **isValid** () const  
*Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.*
- void **setValid** (bool valid)  
*Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.*

### 6.857.1 Detailed Description

Basic type object that holds data that composes a given URI.

### 6.857.2 Constructor & Destructor Documentation

6.857.2.1 `decaf::internal::net::URIType::URIType ( const std::string & source ) [inline]`

6.857.2.2 `decaf::internal::net::URIType::URIType ( ) [inline]`

6.857.2.3 `virtual decaf::internal::net::URIType::~~URIType ( ) [inline, virtual]`

### 6.857.3 Member Function Documentation

6.857.3.1 `std::string decaf::internal::net::URIType::getAuthority ( ) const [inline]`

Gets the Authority of the URI.



**Returns**

Authority part string.

6.857.3.2 `std::string decaf::internal::net::URIType::getFragment ( ) const [inline]`

Gets the Fragment part of the URI.

**Returns**

Fragment part string.

6.857.3.3 `std::string decaf::internal::net::URIType::getHost ( ) const [inline]`

Gets the Host name part of the URI.

**Returns**

Host name part string.

6.857.3.4 `std::string decaf::internal::net::URIType::getPath ( ) const [inline]`

Gets the Path part of the URI.

**Returns**

Path part string.

6.857.3.5 `int decaf::internal::net::URIType::getPort ( ) const [inline]`

Gets the port part of the URI.

**Returns**

port part string, -1 if not set.

6.857.3.6 `std::string decaf::internal::net::URIType::getQuery ( ) const [inline]`

Gets the Query part of the URI.

**Returns**

Query part string.

6.857.3.7 `std::string decaf::internal::net::URIType::getScheme ( ) const [inline]`

Gets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

#### Returns

scheme part string.

6.857.3.8 `std::string decaf::internal::net::URIType::getSchemeSpecificPart ( ) const [inline]`

Gets the Scheme Specific Part of the URI.

#### Returns

scheme specific part string.

6.857.3.9 `std::string decaf::internal::net::URIType::getSource ( ) const [inline]`

Gets the source URI string that was parsed to obtain this **URIType** (p. 3884) instance and the resulting data,.

#### Returns

the source URI string

6.857.3.10 `std::string decaf::internal::net::URIType::getUserInfo ( ) const [inline]`

Gets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

#### Returns

user info part string.

6.857.3.11 `bool decaf::internal::net::URIType::isAbsolute ( ) const [inline]`

Gets if the URI is Absolute.

#### Returns

true if Absolute.

6.857.3.12 `bool decaf::internal::net::URIType::isOpaque ( ) const [inline]`

Gets if the URI is Opaque.

#### Returns

true if opaque.

6.857.3.13 `bool decaf::internal::net::URIType::isServerAuthority ( ) const [inline]`

Gets if the URI is a Server Authority.

#### Returns

true if Server Authority.

6.857.3.14 `bool decaf::internal::net::URIType::isValid ( ) const [inline]`

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

#### Returns

true if the **URIType** (p. 3884) contains valid data.

6.857.3.15 `void decaf::internal::net::URIType::setAbsolute ( bool absolute ) [inline]`

Sets if the URI is Absolute.

#### Parameters

<i>absolute</i>	- true if Absolute.
-----------------	---------------------

6.857.3.16 `void decaf::internal::net::URIType::setAuthority ( const std::string & authority ) [inline]`

Sets the Authority of the URI.

#### Parameters

<i>authority</i>	Authority part string.
------------------	------------------------

6.857.3.17 `void decaf::internal::net::URIType::setFragment ( const std::string & fragment )`  
[inline]

Sets the Fragment part of the URI.

#### Parameters

<i>fragment</i>	- Fragment part string.
-----------------	-------------------------

6.857.3.18 `void decaf::internal::net::URIType::setHost ( const std::string & host )`  
[inline]

Sets the Host name part of the URI.

#### Parameters

<i>host</i>	- Host name part string.
-------------	--------------------------

6.857.3.19 `void decaf::internal::net::URIType::setOpaque ( bool opaque )` [inline]

Sets if the URI is Opaque.

#### Parameters

<i>opaque</i>	true if opaque.
---------------	-----------------

6.857.3.20 `void decaf::internal::net::URIType::setPath ( const std::string & path )`  
[inline]

Sets the Path part of the URI.

#### Parameters

<i>path</i>	- Path part string.
-------------	---------------------

6.857.3.21 `void decaf::internal::net::URIType::setPort ( int port )` [inline]

Sets the port part of the URI.

#### Parameters

<i>port</i>	- port part string, -1 if not set.
-------------	------------------------------------

6.857.3.22 void decaf::internal::net::URIType::setQuery ( const std::string & *query* )  
[inline]

Sets the Query part of the URI.

#### Parameters

<i>query</i>	- Query part string.
--------------	----------------------

6.857.3.23 void decaf::internal::net::URIType::setScheme ( const std::string & *scheme* )  
[inline]

Sets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

#### Parameters

<i>scheme</i>	- scheme part string.
---------------	-----------------------

6.857.3.24 void decaf::internal::net::URIType::setSchemeSpecificPart ( const std::string & *schemeSpecificPart* ) [inline]

Sets the Scheme Specific Part of the URI.

#### Parameters

<i>scheme-SpecificPart</i>	- scheme specific part string.
----------------------------	--------------------------------

6.857.3.25 void decaf::internal::net::URIType::setServerAuthority ( bool *serverAuthority* )  
[inline]

Sets if the URI is a Server Authority.

#### Parameters

<i>server-Authority</i>	- true if Server Authority.
-------------------------	-----------------------------

6.857.3.26 void decaf::internal::net::URIType::setSource ( const std::string & *source* )  
[inline]

Sets the source URI string that was parsed to obtain this **URIType** (p. 3884) instance and the resulting data,.

**Parameters**

<i>source</i>	- the source URI string
---------------	-------------------------

6.857.3.27 `void decaf::internal::net::URIType::setUserInfo ( const std::string & userinfo )`  
`[inline]`

Sets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

**Parameters**

<i>userinfo</i>	- user info part string.
-----------------	--------------------------

6.857.3.28 `void decaf::internal::net::URIType::setValid ( bool valid )` `[inline]`

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

**Parameters**

<i>valid</i>	- true if the <b>URIType</b> (p. 3884) contains valid data.
--------------	---

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIType.h`

**6.858 decaf::net::URL Class Reference**

Class **URL** (p. 3891) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

```
#include <src/main/decaf/net/URL.h>
```

**Public Member Functions**

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

**6.858.1 Detailed Description**

Class **URL** (p. 3891) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

```
http://www.ksc.nasa.gov/facts/internet/url-primer.html
```

In general, a **URL** (p. 3891) can be broken into several parts. The previous example of a **URL** (p. 3891) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named www.ksc.nasa.gov. The information on that host machine is named /facts/internet/url-primer.html. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 3891) is called the path component.

A **URL** (p. 3891) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

```
http://www.ksc.nasa.gov:80/facts/internet/url-primer.html
```

The syntax of **URL** (p. 3891) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 3853)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports scope\_ids. The syntax and usage of scope\_ids is described here.

A **URL** (p. 3891) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

```
http://www.apache.org/cms/index.html#chapter1
```

This fragment is not technically part of the **URL** (p. 3891). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that part of the document that has the tag chapter1 attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 3891). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 3891):

```
http://www.apache.org/cms/index.html
```

contained within it the relative **URL** (p. 3891):

```
FAQ.html
```

it would be a shorthand for:

```
http://www.apache.org/cms/FAQ.html
```

The relative **URL** (p. 3891) need not specify all the components of a **URL** (p. 3891). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 3891). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 3891) class does not itself encode or decode any **URL** (p. 3891) components according to the escaping mechanism defined in RFC2396. It is the responsi-

bility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 3891), and also to decode any escaped fields, that are returned from **URL** (p. 3891). Furthermore, because **URL** (p. 3891) has no knowledge of **URL** (p. 3891) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 3891). For example, the two URLs:

```
http://foo.com/hello world/ and http://foo.com/hello%20world
```

would be considered not equal to each other.

Note, the **URI** (p. 3853) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 3853), and to convert between these two classes using **toURI()** and **URI.toURL()** (p. 3864).

The **URLEncoder** (p. 3894) and **URLDecoder** (p. 3893) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

## 6.858.2 Constructor & Destructor Documentation

6.858.2.1 **decaf::net::URL::URL ( )**

6.858.2.2 **decaf::net::URL::URL ( const std::string & url )**

6.858.2.3 **virtual decaf::net::URL::~~URL ( )** [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URL.h**

## 6.859 decaf::net::URLDecoder Class Reference

```
#include <src/main/decaf/net/URLDecoder.h>
```

### Public Member Functions

- virtual **~URLDecoder ( )**

### Static Public Member Functions

- static std::string **decode** (const std::string &value)

*Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type.*



### 6.859.1 Constructor & Destructor Documentation

6.859.1.1 virtual decaf::net::URLDecoder::~~URLDecoder ( ) [inline, virtual]

### 6.859.2 Member Function Documentation

6.859.2.1 static std::string decaf::net::URLDecoder::decode ( const std::string & value )  
[static]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type.

'+' will be converted to space, '%' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C \$%"

#### Parameters

<i>value</i>	- string The encoded string.
--------------	------------------------------

#### Returns

The decoded version as a string.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URLDecoder.h**

## 6.860 decaf::net::URLEncoder Class Reference

```
#include <src/main/decaf/net/URLEncoder.h>
```

### Public Member Functions

- virtual ~**URLEncoder** ( )

### Static Public Member Functions

- static std::string **encode** (const std::string &value)

*This class contains a utility method for converting a string to the format required by the application/x-www-form-urlencoded MIME content type.*

### 6.860.1 Constructor & Destructor Documentation

6.860.1.1 virtual decaf::net::URLEncoder::~~URLEncoder ( ) [inline, virtual]

## 6.860.2 Member Function Documentation

6.860.2.1 `static std::string decaf::net::URLEncoder::encode ( const std::string & value )`  
`[static]`

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '\*', '\_' are converted into their hexadecimal value prepended by '%'.

For example: '#' -> '%23'

In addition, spaces are substituted by '+'

### Parameters

<code>value</code>	- the string to be converted
--------------------	------------------------------

### Returns

the converted string

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLEncoder.h`

## 6.861 activemq::util::Usage Class Reference

```
#include <src/main/activemq/util/Usage.h>
```

Inheritance diagram for `activemq::util::Usage`:

### Public Member Functions

- virtual `~Usage ()`
- virtual void `waitForSpace ()=0`  
*Waits forever for more space to be returned to this **Usage** (p. 3895) Manager.*
- virtual void `waitForSpace (unsigned int timeout)=0`  
*Waits for more space to be returned to this **Usage** (p. 3895) Manager, times out when the given time span in milliseconds elapses.*
- virtual void `enqueueUsage (unsigned long long value)=0`  
*Tries to increase the usage by value amount but blocks if this object is currently full.*
- virtual void `increaseUsage (unsigned long long value)=0`  
*Increases the usage by the value amount.*
- virtual void `decreaseUsage (unsigned long long value)=0`

*Decreases the usage by the value amount.*

- virtual bool **isFull** () const =0

*Returns true if this **Usage** (p. 3895) instance is full, i.e.*

## 6.861.1 Constructor & Destructor Documentation

6.861.1.1 virtual activemq::util::Usage::~Usage ( ) [inline, virtual]

## 6.861.2 Member Function Documentation

6.861.2.1 virtual void activemq::util::Usage::decreaseUsage ( unsigned long long value )  
[pure virtual]

Decreases the usage by the value amount.

### Parameters

<i>value</i>	Amount of space to return to the pool
--------------	---------------------------------------

Implemented in **activemq::util::MemoryUsage** (p. 2473).

6.861.2.2 virtual void activemq::util::Usage::enqueueUsage ( unsigned long long value )  
[pure virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

### Parameters

<i>value</i>	Amount of usage in bytes to add.
--------------	----------------------------------

Implemented in **activemq::util::MemoryUsage** (p. 2474).

6.861.2.3 virtual void activemq::util::Usage::increaseUsage ( unsigned long long value )  
[pure virtual]

Increases the usage by the value amount.

### Parameters

<i>value</i>	Amount of usage to add.
--------------	-------------------------

Implemented in **activemq::util::MemoryUsage** (p. 2474).

6.861.2.4 virtual bool activemq::util::Usage::isFull ( ) const [pure virtual]

Returns true if this **Usage** (p. 3895) instance is full, i.e.

**Usage** (p. 3895)  $\geq 100\%$

### Returns

true if **Usage** (p. 3895) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 2474).

**6.861.2.5** `virtual void activemq::util::Usage::waitForSpace ( unsigned int timeout ) [pure virtual]`

Waits for more space to be returned to this **Usage** (p. 3895) Manager, times out when the given time span in milliseconds elapses.

### Parameters

<i>timeout</i>	The time to wait for more space.
----------------	----------------------------------

Implemented in **activemq::util::MemoryUsage** (p. 2475).

**6.861.2.6** `virtual void activemq::util::Usage::waitForSpace ( ) [pure virtual]`

Waits forever for more space to be returned to this **Usage** (p. 3895) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 2475).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/Usage.h`

## 6.862 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

```
#include <src/main/decaf/io/UTFDataFormatException.h>
```

Inheritance diagram for `decaf::io::UTFDataFormatException`:

### Public Member Functions

- **UTFDataFormatException** () throw ()  
*Default Constructor.*
- **UTFDataFormatException** (const **lang::Exception** &ex) throw ()  
*Copy Constructor.*

- **UTFDataFormatException** (const **UTFDataFormatException** &ex) throw ()  
*Copy Constructor.*
- **UTFDataFormatException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **UTFDataFormatException** (const std::exception \*cause) throw ()  
*Constructor.*
- **UTFDataFormatException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **UTFDataFormatException** \* **clone** () const  
*Clones this exception.*
- virtual ~**UTFDataFormatException** () throw ()

### 6.862.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

#### Since

1.0

### 6.862.2 Constructor & Destructor Documentation

6.862.2.1 decaf::io::UTFDataFormatException::UTFDataFormatException ( ) throw ()  
[inline]

Default Constructor.

6.862.2.2 decaf::io::UTFDataFormatException::UTFDataFormatException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

#### Parameters

ex	the exception to copy
----	-----------------------

6.862.2.3 decaf::io::UTFDataFormatException::UTFDataFormatException ( const UTFDataFormatException & ex ) throw () [inline]

Copy Constructor.

**Parameters**

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

**6.862.2.4** `decaf::io::UTFDataFormatException::UTFDataFormatException ( const char * file,  
const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()`  
[inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.862.2.5** `decaf::io::UTFDataFormatException::UTFDataFormatException ( const std::exception  
* cause ) throw ()` [inline]

Constructor.

**Parameters**

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

**6.862.2.6** `decaf::io::UTFDataFormatException::UTFDataFormatException ( const char * file,  
const int lineNumber, const char * msg, ... ) throw ()` [inline]

Constructor.

**Parameters**

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
...	list of primitives that are formatted into the message

**6.862.2.7** `virtual decaf::io::UTFDataFormatException::~~UTFDataFormatException ( ) throw ()`  
[inline, virtual]

### 6.862.3 Member Function Documentation

6.862.3.1 virtual **UTFDataFormatException\*** decaf::io::UTFDataFormatException::clone ( ) const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

#### Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**UTFDataFormatException.h**

## 6.863 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 3900)).

```
#include <src/main/decaf/util/UUID.h>
```

Inheritance diagram for decaf::util::UUID:

### Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)  
*Constructs a new **UUID** (p. 3900) using the specified data.*
- virtual **~UUID** ()
- virtual int **compareTo** (const **UUID** &value) const  
*Compare the given **UUID** (p. 3900) to this one.*
- virtual bool **equals** (const **UUID** &value) const  
*Compares this **UUID** (p. 3900) to the one given, returns true if they are equal.*
- virtual bool **operator==** (const **UUID** &value) const  
*Compares equality between this object and the one passed.*
- virtual bool **operator<** (const **UUID** &value) const  
*Compares this object to another and returns true if this object is considered to be less than the one passed.*
- virtual std::string **toString** () const  
*Returns a String object representing this **UUID** (p. 3900).*
- virtual long long **getLeastSignificantBits** () const

- virtual long long **getMostSignificantBits** () const
- virtual long long **node** () throw ( lang::exceptions::UnsupportedOperationException )  
*The node value associated with this **UUID** (p. 3900).*
- virtual long long **timestamp** () throw ( lang::exceptions::UnsupportedOperationException )  
*The timestamp value associated with this **UUID** (p. 3900).*
- virtual int **clockSequence** () throw ( lang::exceptions::UnsupportedOperationException )  
*The clock sequence value associated with this **UUID** (p. 3900).*
- virtual int **variant** () throw ( lang::exceptions::UnsupportedOperationException )  
*The variant number associated with this **UUID** (p. 3900).*
- virtual int **version** () throw ( lang::exceptions::UnsupportedOperationException )  
*The version number associated with this **UUID** (p. 3900).*

### Static Public Member Functions

- static **UUID randomUUID** ()  
*Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3900).*
- static **UUID nameUUIDFromBytes** (const std::vector< char > &name)  
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3900) based on the specified byte array.*
- static **UUID nameUUIDFromBytes** (const char \*name, std::size\_t size)  
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3900) based on the specified byte array.*
- static **UUID fromString** (const std::string &name) throw ( lang::exceptions::IllegalArgumentException )  
*Creates a **UUID** (p. 3900) from the string standard representation as described in the **toString()** (p. 3906) method.*

### 6.863.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 3900)).

A **UUID** (p. 3900) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 3900) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 3900) is as follows: The most significant long consists of the following unsigned fields:

```
0xFFFFFFFF00000000 time_low 0x00000000FFFF0000 time_mid 0x000000000000F000
version 0x000000000000FFF time_hi
```

The least significant long consists of the following unsigned fields:



0xC000000000000000 variant 0x3FFF000000000000 clock\_seq 0x0000FFFFFFFFFFFF node

The variant field contains a value which identifies the layout of the **UUID** (p. 3900). The bit layout described above is valid only for a **UUID** (p. 3900) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 3900). There are four different basic types of UUIDs: time-based, DCE security, name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

## 6.863.2 Constructor & Destructor Documentation

### 6.863.2.1 decaf::util::UUID::UUID ( long long *mostSigBits*, long long *leastSigBits* )

Constructs a new **UUID** (p. 3900) using the specified data.

*mostSigBits* is used for the most significant 64 bits of the **UUID** (p. 3900) and *leastSigBits* becomes the least significant 64 bits of the **UUID** (p. 3900).

#### Parameters

<i>mostSigBits</i>	
<i>leastSigBits</i>	

### 6.863.2.2 virtual decaf::util::UUID::~~UUID ( ) [virtual]

## 6.863.3 Member Function Documentation

### 6.863.3.1 virtual int decaf::util::UUID::clockSequence ( ) throw ( lang::exceptions::UnsupportedOperationException ) [virtual]

The clock sequence value associated with this **UUID** (p. 3900).

The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 3900). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 3900).

The *clockSequence* value is only meaningful in a time-based **UUID** (p. 3900), which has version type 1. If this **UUID** (p. 3900) is not a time-based **UUID** (p. 3900) then this method throws `UnsupportedOperationException`.

#### Returns

the *clockSequence* associated with a V1 **UUID** (p. 3900)

**Exceptions**

<i>UnsupportedOperationException</i>
--------------------------------------

6.863.3.2 `virtual int decaf::util::UUID::compareTo ( const UUID & value ) const`  
[virtual]

Compare the given **UUID** (p. 3900) to this one.

**Parameters**

<i>value</i>	- the <b>UUID</b> (p. 3900) to compare to
--------------	---

6.863.3.3 `virtual bool decaf::util::UUID::equals ( const UUID & value ) const` [virtual]

Compares this **UUID** (p. 3900) to the one given, returns true if they are equal.

**Parameters**

<i>value</i>	- the <b>UUID</b> (p. 3900) to compare to.
--------------	--

**Returns**

true if UUIDs are the same.

6.863.3.4 `static UUID decaf::util::UUID::fromString ( const std::string & name ) throw (`  
`lang::exceptions::IllegalArgumentException )` [static]

Creates a **UUID** (p. 3900) from the string standard representation as described in the **toString()** (p. 3906) method.

**Parameters**

<i>name</i>	- a string to be used to construct a <b>UUID</b> (p. 3900).
-------------	---

**Returns**

type 3 **UUID** (p. 3900)

6.863.3.5 `virtual long long decaf::util::UUID::getLeastSignificantBits ( ) const` [virtual]

**Returns**

the most significant 64 bits of this UUID's 128 bit value.

6.863.3.6 virtual long long decaf::util::UUID::getMostSignificantBits ( ) const [virtual]

#### Returns

the most significant 64 bits of this UUID's 128 bit value.

6.863.3.7 static **UUID** decaf::util::UUID::nameUUIDFromBytes ( const std::vector< char > & name ) [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 3900) based on the specified byte array.

#### Parameters

<i>name</i>	- a byte array to be used to construct a <b>UUID</b> (p. 3900).
-------------	---

#### Returns

type 3 **UUID** (p. 3900)

6.863.3.8 static **UUID** decaf::util::UUID::nameUUIDFromBytes ( const char \* name, std::size\_t size ) [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 3900) based on the specified byte array.

#### Parameters

<i>name</i>	- a byte array to be used to construct a <b>UUID</b> (p. 3900).
<i>size</i>	- the size of the byte array, or number of bytes to use.

#### Returns

type 3 **UUID** (p. 3900)

6.863.3.9 virtual long long decaf::util::UUID::node ( ) throw ( lang::exceptions::UnsupportedOperationException ) [virtual]

The node value associated with this **UUID** (p. 3900).

The 48 bit node value is constructed from the node field of this **UUID** (p. 3900). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 3900) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 3900), which has version type 1. If this **UUID** (p. 3900) is not a time-based **UUID** (p. 3900) then this method throws UnsupportedOperationException.

**Returns**

the node value of this **UUID** (p. 3900)

**Exceptions**

<i>UnsupportedOperation</i> <i>Exception</i>
---

**6.863.3.10** `virtual bool decaf::util::UUID::operator< ( const UUID & value ) const`  
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

**6.863.3.11** `virtual bool decaf::util::UUID::operator== ( const UUID & value ) const`  
[virtual]

Compares equality between this object and the one passed.

**Parameters**

<i>value</i>	- the value to be compared to this one.
--------------	---

**Returns**

true if this object is equal to the one passed.

**6.863.3.12** `static UUID decaf::util::UUID::randomUUID ( )` [static]

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3900).

The **UUID** (p. 3900) is generated using a cryptographically strong pseudo random number generator.

**Returns**

type 4 **UUID** (p. 3900)

6.863.3.13 `virtual long long decaf::util::UUID::timestamp ( ) throw ( lang::exceptions::UnsupportedOperationException ) [virtual]`

The timestamp value associated with this **UUID** (p. 3900).

The 60 bit timestamp value is constructed from the `time_low`, `time_mid`, and `time_hi` fields of this **UUID** (p. 3900). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 3900), which has version type 1. If this **UUID** (p. 3900) is not a time-based **UUID** (p. 3900) then this method throws `UnsupportedOperationException`.

#### Returns

the timestamp associated with a V1 **UUID** (p. 3900)

#### Exceptions

<i>UnsupportedOperationException</i>	
--------------------------------------	--

6.863.3.14 `virtual std::string decaf::util::UUID::toString ( ) const [virtual]`

Returns a `String` object representing this **UUID** (p. 3900).

UUID's are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

#### Returns

formatted string for this **UUID** (p. 3900)

6.863.3.15 `virtual int decaf::util::UUID::variant ( ) throw ( lang::exceptions::UnsupportedOperationException ) [virtual]`

The variant number associated with this **UUID** (p. 3900).

The variant number describes the layout of the **UUID** (p. 3900). The variant number has the following meaning:

\* 0 Reserved for NCS backward compatibility \* 2 The Leach-Salz variant (used by this class) \* 6 Reserved, Microsoft Corporation backward compatibility \* 7 Reserved for future definition

#### Returns

the variant associated with a V1 **UUID** (p. 3900)

#### Exceptions

<i>UnsupportedOperationException</i>	
--------------------------------------	--

6.863.3.16 `virtual int decaf::util::UUID::version ( ) throw ( lang::exceptions::UnsupportedOperationException ) [virtual]`

The version number associated with this **UUID** (p. 3900).

The version number describes how this **UUID** (p. 3900) was generated. The version number has the following meaning:

\* 1 Time-based **UUID** (p. 3900) \* 2 DCE security **UUID** (p. 3900) \* 3 Name-based **UUID** (p. 3900) \* 4 Randomly generated **UUID** (p. 3900)

### Returns

the version associated with a V1 **UUID** (p. 3900)

### Exceptions

<i>UnsupportedOperationException</i>
--------------------------------------

The documentation for this class was generated from the following file:

- src/main/decaf/util/**UUID.h**

## 6.864 activemq::wireformat::WireFormat Class Reference

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

```
#include <src/main/activemq/wireformat/WireFormat.h>
```

Inheritance diagram for activemq::wireformat::WireFormat:

### Public Member Functions

- virtual **~WireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** \*transport, **decaf::io::DataOutputStream** \*out)=0 throw ( **decaf::io::IOException** )  
*Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.*
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** \*transport, **decaf::io::DataInputStream** \*in)=0 throw ( **decaf::io::IOException** )  
*Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.*
- virtual void **setVersion** (int version)=0  
*Set the Version.*

- virtual int **getVersion** () const =0  
*Get the Version.*
- virtual bool **hasNegotiator** () const =0  
*Returns true if this **WireFormat** (p. 3907) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual bool **inReceive** () const =0  
*Indicates if the WireFormat object is in the process of receiving a message.*
- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)=0 throw ( decaf::lang::exceptions::UnsupportedOperationException )  
*If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.*

### 6.864.1 Detailed Description

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

#### Version

#### Revision:

1.1

### 6.864.2 Constructor & Destructor Documentation

6.864.2.1 virtual activemq::wireformat::WireFormat::~WireFormat ( ) [inline, virtual]

### 6.864.3 Member Function Documentation

6.864.3.1 virtual **Pointer**<transport::Transport> activemq::wireformat::WireFormat::createNegotiator ( const **Pointer**< transport::Transport > & transport ) throw ( decaf::lang::exceptions::UnsupportedOperationException ) [pure virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

#### Parameters

<i>transport</i>	- the Transport to Wrap the Negotiator around.
------------------	--

**Returns**

new instance of a **WireFormatNegotiator** (p. 3946) as a **Pointer<Transport>** (p. 2896).

**Exceptions**

<i>UnsupportedOperationException</i>	if the <b>WireFormat</b> (p. 3907) doesn't have a Negotiator.
--------------------------------------	---

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2840), and **activemq::wireformat::stomp::StompWireFormat** (p. 3587).

6.864.3.2 `virtual int activemq::wireformat::WireFormat::getVersion ( ) const` [pure virtual]

Get the Version.

**Returns**

the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2842), and **activemq::wireformat::stomp::StompWireFormat** (p. 3587).

6.864.3.3 `virtual bool activemq::wireformat::WireFormat::hasNegotiator ( ) const` [pure virtual]

Returns true if this **WireFormat** (p. 3907) has a Negotiator that needs to wrap the Transport that uses it.

**Returns**

true if the **WireFormat** (p. 3907) provides a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2842), and **activemq::wireformat::stomp::StompWireFormat** (p. 3587).

6.864.3.4 `virtual bool activemq::wireformat::WireFormat::inReceive ( ) const` [pure virtual]

Indicates if the WireFormat object is in the process of receiving a message.

This is useful for monitoring inactivity and the **WireFormat** (p. 3907) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 3907) instance to determine if its busy or not and not mark the connection as inactive if so.

**Returns**

true if the **WireFormat** (p. 3907) object is unmarshaling a message.



Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2842), and **activemq::wireformat::stomp::StompWireFormat** (p. 3588).

6.864.3.5 `virtual void activemq::wireformat::WireFormat::marshal ( const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out ) throw ( decaf::io::IOException ) [pure virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

#### Parameters

<i>command</i>	The Command to Marshal
<i>transport</i>	The Transport that called this method.
<i>out</i>	The output stream to write the command to.

#### Exceptions

<i>IOException</i>	
--------------------	--

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2844), and **activemq::wireformat::stomp::StompWireFormat** (p. 3588).

6.864.3.6 `virtual void activemq::wireformat::WireFormat::setVersion ( int version ) [pure virtual]`

Set the Version.

#### Parameters

<i>version</i>	the version of the wire format
----------------	--------------------------------

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2847), and **activemq::wireformat::stomp::StompWireFormat** (p. 3588).

6.864.3.7 `virtual Pointer<commands::Command> activemq::wireformat::WireFormat::unmarshal ( const activemq::transport::Transport * transport, decaf::io::DataInputStream * in ) throw ( decaf::io::IOException ) [pure virtual]`

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

#### Parameters

<i>transport</i>	- Pointer to the transport that is making this request.
<i>in</i>	- the input stream to read the command from.

**Returns**

the newly marshaled Command, caller owns the pointer

**Exceptions**

<i>IOException</i>
--------------------

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2849), and **activemq::wireformat::stomp::StompWireFormat** (p. 3589).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormat.h**

## 6.865 activemq::wireformat::WireFormatFactory Class Reference

The **WireFormatFactory** (p. 3911) is the interface that all **WireFormatFactory** (p. 3911) classes must extend.

```
#include <src/main/activemq/wireformat/WireFormatFactory.h>
```

Inheritance diagram for activemq::wireformat::WireFormatFactory:

**Public Member Functions**

- virtual **~WireFormatFactory** ()
- virtual **Pointer< WireFormat > createWireFormat** (const **decaf::util::Properties** &properties)=0 throw ( **decaf::lang::exceptions::IllegalStateException** )

*Creates a new **WireFormat** (p. 3907) Object passing it a set of properties from which it can obtain any optional settings.*

### 6.865.1 Detailed Description

The **WireFormatFactory** (p. 3911) is the interface that all **WireFormatFactory** (p. 3911) classes must extend.

The Factory creates a **WireFormat** (p. 3907) Object based on the properties that are set in the passed **Properties** object.

### 6.865.2 Constructor & Destructor Documentation

- 6.865.2.1 **virtual activemq::wireformat::WireFormatFactory::~WireFormatFactory** ( )  
[inline, virtual]

### 6.865.3 Member Function Documentation

6.865.3.1 virtual **Pointer**<**WireFormat**> **activemq::wireformat::WireFormatFactory::createWireFormat** ( const **decaf::util::Properties** & *properties* ) throw ( **decaf::lang::exceptions::IllegalStateException** ) [pure virtual]

Creates a new **WireFormat** (p. 3907) Object passing it a set of properties from which it can obtain any optional settings.

#### Parameters

<i>properties</i>	- the Properties for this <b>WireFormat</b> (p. 3907)
-------------------	---

#### Returns

Pointer to a new instance of a **WireFormat** (p. 3907) object.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 2850), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 3590).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatFactory.h**

## 6.866 activemq::commands::WireFormatInfo Class Reference

```
#include <src/main/activemq/commands/WireFormatInfo.h>
```

Inheritance diagram for **activemq::commands::WireFormatInfo**:

### Public Member Functions

- **WireFormatInfo** ()
- virtual ~**WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **DataStructure** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this objects members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual bool **isMarshalAware** () const  
*Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.*
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** \*visitor) throw ( exceptions::ActiveMQException )  
*Allows a Visitor to visit this command and return a response to the command based on the command type being visited.*
- int **getVersion** () const  
*Get the current Wireformat Version.*
- void **setVersion** (int version)  
*Set the current Wireformat Version.*
- long long **getMaxInactivityDuration** () const  
*Returns the currently configured Max Inactivity duration.*
- void **setMaxInactivityDuration** (long long maxInactivityDuration)  
*Sets the Max inactivity duration value.*
- long long **getMaxInactivityDurationInitialDelay** () const  
*Returns the currently configured Max Inactivity Initial Delay duration.*
- void **setMaxInactivityDurationInitialDelay** (long long maxInactivityDurationInitialDelay)  
*Sets the Max inactivity initial delay duration value.*
- bool **isStackTraceEnabled** () const  
*Checks if the stackTraceEnabled flag is on.*
- void **setStackTraceEnabled** (bool stackTraceEnabled)  
*Sets if the stackTraceEnabled flag is on.*
- bool **isTcpNoDelayEnabled** () const  
*Checks if the tcpNoDelayEnabled flag is on.*
- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)  
*Sets if the tcpNoDelayEnabled flag is on.*
- bool **isCacheEnabled** () const  
*Checks if the cacheEnabled flag is on.*
- void **setCacheEnabled** (bool cacheEnabled)  
*Sets if the cacheEnabled flag is on.*
- int **getCacheSize** () const  
*Gets the Cache Size setting.*
- void **setCacheSize** (int value)  
*Sets the Cache Size setting.*
- bool **isTightEncodingEnabled** () const  
*Checks if the tightEncodingEnabled flag is on.*
- void **setTightEncodingEnabled** (bool tightEncodingEnabled)  
*Sets if the tightEncodingEnabled flag is on.*
- bool **isSizePrefixDisabled** () const

- Checks if the sizePrefixDisabled flag is on.*
- void **setSizePrefixDisabled** (bool sizePrefixDisabled)  
*Sets if the sizePrefixDisabled flag is on.*
- const std::vector< unsigned char > & **getMagic** () const  
*Get the Magic field.*
- void **setMagic** (const std::vector< unsigned char > &magic)  
*Sets the value of the magic field.*
- const std::vector< unsigned char > & **getMarshaledProperties** () const  
*Get the marshalledProperties field.*
- void **setMarshaledProperties** (const std::vector< unsigned char > &marshalled-Properties)  
*Sets the value of the marshalledProperties field.*
- virtual const **util::PrimitiveMap** & **getProperties** () const  
*Gets the Properties for this **Command** (p. 1165).*
- virtual **util::PrimitiveMap** & **getProperties** ()  
*Gets the Properties for this **Command** (p. 1165).*
- virtual void **setProperties** (const **util::PrimitiveMap** &map)  
*Sets the Properties for this **Command** (p. 1165).*
- bool **isValid** () const  
*Determines if we think this is a Valid **WireFormatInfo** (p. 3912) command.*
- virtual bool **isWireFormatInfo** () const
- virtual void **beforeMarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.*
- virtual void **afterUnmarshal** (**wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED) throw ( decaf::io::IOException )  
*Called after unmarshaling is started to cleanup the object being unmarshaled.*

## Static Public Attributes

- static const unsigned char **ID\_WIREFORMATINFO** = 1

## 6.866.1 Constructor & Destructor Documentation

6.866.1.1 **activemq::commands::WireFormatInfo::WireFormatInfo** ( )

6.866.1.2 **virtual activemq::commands::WireFormatInfo::~WireFormatInfo** ( ) [virtual]

## 6.866.2 Member Function Documentation

6.866.2.1 **virtual void activemq::commands::WireFormatInfo::afterUnmarshal** ( **wireformat::WireFormat** \*wireFormat AMQCPP\_UNUSED ) throw ( decaf::io::IOException ) [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

**Parameters**

<i>wireFormat</i>	- the wireformat object to control unmarshaling
-------------------	---

Reimplemented from **activemq::commands::BaseDataStructure** (p. 794).

```
6.866.2.2  virtual void activemq::commands::WireFormatInfo::beforeMarshal (
            wireformat::WireFormat *wireFormat AMQCPP_UNUSED ) throw (
            decaf::io::IOException ) [virtual]
```

Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

**Parameters**

<i>wireFormat</i>	- the wire formatting controller
-------------------	----------------------------------

Reimplemented from **activemq::commands::BaseDataStructure** (p. 794).

```
6.866.2.3  virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure (
            ) const [virtual]
```

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

**Returns**

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1628).

```
6.866.2.4  virtual void activemq::commands::WireFormatInfo::copyDataStructure ( const
            DataStructure * src ) [virtual]
```

Copy the contents of the passed object into this objects members, overwriting any existing data.

**Returns**

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 724).

```
6.866.2.5  virtual bool activemq::commands::WireFormatInfo::equals ( const DataStructure *
            value ) const [virtual]
```

Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

### Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 725).

6.866.2.6 `int activemq::commands::WireFormatInfo::getCacheSize ( ) const`

Gets the Cache Size setting.

### Returns

currently set cache size.

6.866.2.7 `virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

### Returns

new **DataStructure** (p. 1628) type copy.

Implements **activemq::commands::DataStructure** (p. 1631).

6.866.2.8 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic ( ) const [inline]`

Get the Magic field.

### Returns

const reference to a `std::vector<char>`

6.866.2.9 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshaledProperties ( ) const [inline]`

Get the marshalledProperties field.

### Returns

const reference to a `std::vector<char>`

6.866.2.10 `long long activemq::commands::WireFormatInfo::getMaxInactivityDuration ( ) const`

Returns the currently configured Max Inactivity duration.

#### Returns

the set inactivity duration value.

6.866.2.11 `long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitialDelay ( ) const`

Returns the currently configured Max Inactivity Initial Delay duration.

#### Returns

the set inactivity duration initial delay value.

6.866.2.12 `virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ( ) [inline, virtual]`

Gets the Properties for this **Command** (p. 1165).

#### Returns

the Properties object for this **Command** (p. 1165).

6.866.2.13 `virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ( ) const [inline, virtual]`

Gets the Properties for this **Command** (p. 1165).

#### Returns

the Properties object for this **Command** (p. 1165).

6.866.2.14 `int activemq::commands::WireFormatInfo::getVersion ( ) const [inline]`

Get the current Wireformat Version.

#### Returns

int that identifies the version



6.866.2.15 `bool activemq::commands::WireFormatInfo::isCacheEnabled ( ) const`

Checks if the cacheEnabled flag is on.

#### Returns

true if the flag is on.

6.866.2.16 `virtual bool activemq::commands::WireFormatInfo::isMarshalAware ( ) const`  
`[inline, virtual]`

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

#### Returns

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 796).

6.866.2.17 `bool activemq::commands::WireFormatInfo::isSizePrefixDisabled ( ) const`

Checks if the sizePrefixDisabled flag is on.

#### Returns

true if the flag is on.

6.866.2.18 `bool activemq::commands::WireFormatInfo::isStackTraceEnabled ( ) const`

Checks if the stackTraceEnabled flag is on.

#### Returns

true if the flag is on.

6.866.2.19 `bool activemq::commands::WireFormatInfo::isTcpNoDelayEnabled ( ) const`

Checks if the tcpNoDelayEnabled flag is on.

#### Returns

true if the flag is on.

6.866.2.20 `bool activemq::commands::WireFormatInfo::isTightEncodingEnabled ( ) const`

Checks if the `tightEncodingEnabled` flag is on.

#### Returns

true if the flag is on.

6.866.2.21 `bool activemq::commands::WireFormatInfo::isValid ( ) const`

Determines if we think this is a Valid **WireFormatInfo** (p. 3912) command.

#### Returns

true if its valid.

6.866.2.22 `virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo ( ) const`  
[inline, virtual]

#### Returns

answers true to the `isWireFormatInfo` query

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.866.2.23 `void activemq::commands::WireFormatInfo::setCacheEnabled ( bool cacheEnabled )`

Sets if the `cacheEnabled` flag is on.

#### Parameters

<i>cacheEnabled</i>	- true to turn flag is on
---------------------	---------------------------

6.866.2.24 `void activemq::commands::WireFormatInfo::setCacheSize ( int value )`

Sets the Cache Size setting.

#### Parameters

<i>value</i>	- value to set to the cache size.
--------------	-----------------------------------

6.866.2.25 void activemq::commands::WireFormatInfo::setMagic ( const std::vector< unsigned char > & *magic* ) [inline]

Sets the value of the magic field.

#### Parameters

<i>magic</i>	- const std::vector<char>
--------------	---------------------------

6.866.2.26 void activemq::commands::WireFormatInfo::setMarshaledProperties ( const std::vector< unsigned char > & *marshalledProperties* ) [inline]

Sets the value of the marshalledProperties field.

#### Parameters

<i>marshalled-Properties</i>	The Byte Array vector that contains the marshaled form of the <b>Message</b> (p. 2475) properties, this is the data sent over the wire.
------------------------------	---

6.866.2.27 void activemq::commands::WireFormatInfo::setMaxInactivityDuration ( long long *maxInactivityDuration* )

Sets the Max inactivity duration value.

#### Parameters

<i>maxInactivityDuration</i>	- max time a client can be inactive.
------------------------------	--------------------------------------

6.866.2.28 void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitalDelay ( long long *maxInactivityDurationInitalDelay* )

Sets the Max inactivity initial delay duration value.

#### Parameters

<i>maxInactivityDurationInitalDelay</i>	- time before the inactivity delay is checked.
---	--

6.866.2.29 virtual void activemq::commands::WireFormatInfo::setProperties ( const util::PrimitiveMap & *map* ) [inline, virtual]

Sets the Properties for this **Command** (p. 1165).

**Parameters**

<i>map</i>	- PrimitiveMap to copy
------------	------------------------

6.866.2.30 `void activemq::commands::WireFormatInfo::setSizePrefixDisabled ( bool  
sizePrefixDisabled )`

Sets if the sizePrefixDisabled flag is on.

**Parameters**

<i>sizePre- fixDisabled</i>	- true to turn flag is on
---------------------------------	---------------------------

6.866.2.31 `void activemq::commands::WireFormatInfo::setStackTraceEnabled ( bool  
stackTraceEnabled )`

Sets if the stackTraceEnabled flag is on.

**Parameters**

<i>stack- TraceEn- abled</i>	- ture to turn flag is on
--------------------------------------	---------------------------

6.866.2.32 `void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled ( bool  
tcpNoDelayEnabled )`

Sets if the tcpNoDelayEnabled flag is on.

**Parameters**

<i>tcpNoDe- layEnabled</i>	- ture to turn flag is on
--------------------------------	---------------------------

6.866.2.33 `void activemq::commands::WireFormatInfo::setTightEncodingEnabled ( bool  
tightEncodingEnabled )`

Sets if the tightEncodingEnabled flag is on.

**Parameters**

<i>tightEn- codingEn- abled</i>	- true to turn flag is on
---	---------------------------

6.866.2.34 `void activemq::commands::WireFormatInfo::setVersion ( int version )`  
[inline]

Set the current Wireformat Version.

#### Parameters

<i>version</i>	- int that identifies the version
----------------	-----------------------------------

6.866.2.35 `virtual std::string activemq::commands::WireFormatInfo::toString ( ) const`  
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 729).

6.866.2.36 `virtual decaf::lang::Pointer<commands::Command>`  
`activemq::commands::WireFormatInfo::visit (`  
`activemq::state::CommandVisitor * visitor ) throw (`  
`exceptions::ActiveMQException )` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

#### Returns

a **Response** (p. 3227) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 1170).

### 6.866.3 Field Documentation

6.866.3.1 `const unsigned char activemq::commands::WireFormatInfo::ID_`  
`WIREFORMATINFO = 1` [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**WireFormatInfo.h**

## 6.867 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3923).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.867.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3923).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.867.2 Constructor & Destructor Documentation

6.867.2.1 `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]`

6.867.2.2 `virtual activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ( ) [inline, virtual]`

## 6.867.3 Member Function Documentation

6.867.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.867.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.867.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.867.3.4  virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.867.3.5  virtual int activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).



6.867.3.6 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal2  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
 decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
 ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.867.3.7 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightUnmarshal  
 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**WireFormatInfoMarshaller.h**

## 6.868 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3927).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.868.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3927).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.868.2 Constructor & Destructor Documentation

6.868.2.1 `activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]`

6.868.2.2 `virtual activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ( ) [inline, virtual]`

## 6.868.3 Member Function Documentation

6.868.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.868.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.868.3.3 `virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.868.3.4  virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.868.3.5  virtual int activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

## 6.868 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller

### Class Reference 3943

6.868.3.6 virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.868.3.7 virtual void activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/**WireFormatInfoMarshaller.h**

## 6.869 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3931).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.869.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3931).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.869.2 Constructor & Destructor Documentation

6.869.2.1 `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]`

6.869.2.2 `virtual activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ( ) [inline, virtual]`

## 6.869.3 Member Function Documentation

6.869.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.869.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.869.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.869.3.4  virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.869.3.5  virtual int activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).



## 6.869 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller

### Class Reference 3947

6.869.3.6 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.869.3.7 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h

## 6.870 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3935).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.870.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3935).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.870.2 Constructor & Destructor Documentation

6.870.2.1 `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]`

6.870.2.2 `virtual activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ( ) [inline, virtual]`

## 6.870.3 Member Function Documentation

6.870.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.870.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.870.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.870.3.4  virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.870.3.5  virtual int activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

## 6.870 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller

### Class Reference 3951

6.870.3.6 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.870.3.7 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**WireFormatInfoMarshaller.h**

## 6.871 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3939).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.871.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3939).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.871.2 Constructor & Destructor Documentation

6.871.2.1 `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]`

6.871.2.2 `virtual activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ( ) [inline, virtual]`

## 6.871.3 Member Function Documentation

6.871.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.871.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.871.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.871.3.4  virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.871.3.5  virtual int activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).



## 6.871 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller

### Class Reference 3955

6.871.3.6 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.871.3.7 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**WireFormatInfoMarshaller.h**

## 6.872 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller

### Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3943).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller:

### Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.872.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3943).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.872.2 Constructor & Destructor Documentation

6.872.2.1 `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::WireFormatInfoMarshaller ( ) [inline]`

6.872.2.2 `virtual activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller ( ) [inline, virtual]`

## 6.872.3 Member Function Documentation

6.872.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.872.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.872.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1591).

```
6.872.3.4  virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1599).

```
6.872.3.5  virtual int activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1606).

6.872.3.6 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal2 ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1613).

6.872.3.7 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightUnmarshal ( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \* *bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshal an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1620).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**

## 6.873 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 3946) which allows a **WireFormat** (p. 3907) to.

```
#include <src/main/activemq/wireformat/WireFormatNegotiator.h>
```

Inheritance diagram for `activemq::wireformat::WireFormatNegotiator`:

## Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next)  
*Constructor.*
- virtual ~**WireFormatNegotiator** ()

### 6.873.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 3946) which allows a **WireFormat** (p. 3907) to.

### 6.873.2 Constructor & Destructor Documentation

6.873.2.1 `activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator ( const Pointer< transport::Transport > &next ) [inline]`

Constructor.

#### Parameters

<i>next</i>	- the next Transport in the chain
-------------	-----------------------------------

6.873.2.2 `virtual activemq::wireformat::WireFormatNegotiator::~~WireFormatNegotiator ( ) [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatNegotiator.h`

## 6.874 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 3907) Factories that are available to the client at runtime.

```
#include <src/main/activemq/wireformat/WireFormatRegistry.h>
```

## Public Member Functions

- virtual ~**WireFormatRegistry** ()

- **WireFormatFactory** \* **findFactory** (const std::string &name) const throw ( decaf::lang::exceptions::NoSuchElementException )  
*Gets a Registered **WireFormatFactory** (p. 3911) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*
- void **registerFactory** (const std::string &name, **WireFormatFactory** \*factory) throw ( decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException )  
*Registers a new **WireFormatFactory** (p. 3911) with this Registry.*
- void **unregisterFactory** (const std::string &name)  
*Unregisters the Factory with the given name and deletes that instance of the Factory.*
- std::vector< std::string > **getWireFormatNames** () const  
*Retrieves a list of the names of all the Registered WireFormat's in this Registry.*

### Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()  
*Gets the single instance of the **WireFormatRegistry** (p. 3947).*

### 6.874.1 Detailed Description

Registry of all **WireFormat** (p. 3907) Factories that are available to the client at runtime.

New WireFormat's must have a factory registered here before a connection attempt is made.

#### Since

3.0

### 6.874.2 Constructor & Destructor Documentation

6.874.2.1 virtual activemq::wireformat::WireFormatRegistry::~WireFormatRegistry ( )  
 [virtual]

### 6.874.3 Member Function Documentation

6.874.3.1 **WireFormatFactory**\* activemq::wireformat::WireFormatRegistry::findFactory  
 ( const std::string & name ) const throw ( decaf::lang::exceptions::NoSuchElementException )

Gets a Registered **WireFormatFactory** (p. 3911) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

#### Parameters

<i>name</i>	The name of the Factory to find in the Registry.
-------------	--

**Returns**

the Factory registered under the given name.

**Exceptions**

<i>NoSuchElementException</i>	if no factory is registered with that name.
-------------------------------	---

```
6.874.3.2 static WireFormatRegistry& activemq::wireformat::WireFormatRegistry::getInstance ( )  
[static]
```

Gets the single instance of the **WireFormatRegistry** (p. 3947).

**Returns**

reference to the single instance of this Registry

```
6.874.3.3 std::vector<std::string> activemq::wireformat::WireFormatRegistry::getWireFormatNames  
( ) const
```

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

**Returns**

stl vector of strings with all the **WireFormat** (p. 3907) names registered.

```
6.874.3.4 void activemq::wireformat::WireFormatRegistry::registerFactory  
( const std::string & name, WireFormatFactory * factory )  
throw ( decaf::lang::exceptions::IllegalArgumentException,  
decaf::lang::exceptions::NullPointerException )
```

Registers a new **WireFormatFactory** (p. 3911) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

**Parameters**

<i>name</i>	The name of the new Factory to register.
<i>factory</i>	The new Factory to add to the Registry.

**Exceptions**

<i>IllegalArgumentException</i>	is name is the empty string.
---------------------------------	------------------------------



<i>NullPointerException</i>	if the Factory is Null.
-----------------------------	-------------------------

6.874.3.5 void activemq::wireformat::WireFormatRegistry::unregisterFactory ( const std::string & name )

Unregisters the Factory with the given name and deletes that instance of the Factory.

#### Parameters

<i>name</i>	Name of the Factory to unregister and destroy
-------------	---

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatRegistry.h**

## 6.875 activemq::transport::inactivity::WriteChecker Class Reference

Runnable class used by the {.

```
#include <src/main/activemq/transport/inactivity/WriteChecker.h>
```

Inheritance diagram for activemq::transport::inactivity::WriteChecker:

### Public Member Functions

- **WriteChecker** (**InactivityMonitor** \*parent)
- virtual ~**WriteChecker** ()
- virtual void **run** ()

*Run method - called by the Thread class in the context of the thread.*

### 6.875.1 Detailed Description

Runnable class used by the {.

#### See also

**InactivityMonitor** (p. 1964)} to make periodic writes to the underlying **transport** (p. 99) if no other write activity is going on in order to more quickly detect failures of the connection to the broker.

#### Since

3.1.0

## 6.875.2 Constructor & Destructor Documentation

6.875.2.1 `activemq::transport::inactivity::WriteChecker::WriteChecker ( InactivityMonitor * parent )`

6.875.2.2 `virtual activemq::transport::inactivity::WriteChecker::~~WriteChecker ( )`  
[virtual]

## 6.875.3 Member Function Documentation

6.875.3.1 `virtual void activemq::transport::inactivity::WriteChecker::run ( )` [virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 3265).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/WriteChecker.h`

## 6.876 decaf::io::Writer Class Reference

```
#include <src/main/decaf/io/Writer.h>
```

Inheritance diagram for `decaf::io::Writer`:

### Public Member Functions

- **Writer** ()
- virtual **~Writer** ()
- virtual void **write** (char v) throw ( decaf::io::IOException )  
*Writes an single byte char value.*
- virtual void **write** (const std::vector< char > &buffer) throw ( decaf::io::IOException )  
*Writes an array of Chars.*
- virtual void **write** (const char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )  
*Writes a byte array to the output stream.*
- virtual void **write** (const char \*buffer, int size, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes a byte array to the output stream.*
- virtual void **write** (const std::string &str) throw ( decaf::io::IOException )  
*Writes a string.*

- virtual void **write** (const std::string &str, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Writes a string.*
- virtual **decaf::lang::Appendable & append** (char value) throw ( decaf::io::IOException )  
*Appends the specified character to this Appendable.*
- virtual **decaf::lang::Appendable & append** (const **decaf::lang::CharSequence** \*csq) throw ( decaf::io::IOException )  
*Appends the specified character sequence to this Appendable.*
- virtual **decaf::lang::Appendable & append** (const **decaf::lang::CharSequence** \*csq, int start, int end) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Appends a subsequence of the specified character sequence to this Appendable.*

## Protected Member Functions

- virtual void **doWriteArrayBounded** (const char \*buffer, int size, int offset, int length)=0 throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )  
*Override this method to customize the functionality of the method write( char\* buffer, int size, int offset, int length ).*
- virtual void **doWriteChar** (char v) throw ( decaf::io::IOException )
- virtual void **doWriteVector** (const std::vector< char > &buffer) throw ( decaf::io::IOException )
- virtual void **doWriteArray** (const char \*buffer, int size) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException )
- virtual void **doWriteString** (const std::string &str) throw ( decaf::io::IOException )
- virtual void **doWriteStringBounded** (const std::string &str, int offset, int length) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )
- virtual **decaf::lang::Appendable & doAppendChar** (char value) throw ( decaf::io::IOException )
- virtual **decaf::lang::Appendable & doAppendCharSequence** (const **decaf::lang::CharSequence** \*csq) throw ( decaf::io::IOException )
- virtual **decaf::lang::Appendable & doAppendCharSequenceStartEnd** (const **decaf::lang::CharSequence** \*csq, int start, int end) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )

## 6.876.1 Constructor & Destructor Documentation

6.876.1.1 decaf::io::Writer::Writer ( )

6.876.1.2 virtual decaf::io::Writer::~Writer ( ) [virtual]

## 6.876.2 Member Function Documentation

6.876.2.1 `virtual decaf::lang::Appendable& decaf::io::Writer::append ( char value ) throw ( decaf::io::IOException )` [virtual]

Appends the specified character to this Appendable.

#### Parameters

<i>value</i>	The character to append.
--------------	--------------------------

#### Returns

a Reference to this Appendable

#### Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

Implements **decaf::lang::Appendable** (p. 694).

6.876.2.2 `virtual decaf::lang::Appendable& decaf::io::Writer::append ( const decaf::lang::CharSequence * csq ) throw ( decaf::io::IOException )` [virtual]

Appends the specified character sequence to this Appendable.

#### Parameters

<i>csq</i>	The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
------------	---

#### Returns

a Reference to this Appendable.

#### Exceptions

<i>Exception</i>	if an error occurs.
------------------	---------------------

Implements **decaf::lang::Appendable** (p. 695).

6.876.2.3 `virtual decaf::lang::Appendable& decaf::io::Writer::append ( const decaf::lang::CharSequence * csq, int start, int end ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Appends a subsequence of the specified character sequence to this Appendable.

#### Parameters

<i>csq</i>	- The character sequence from which a subsequence will be appended. If <i>csq</i> is NULL, then characters will be appended as if <i>csq</i> contained the string "null".
<i>start</i>	The index of the first character in the subsequence.
<i>end</i>	The index of the character following the last character in the subsequence.

**Returns**

a Reference to this Appendable

**Exceptions**

<i>Exception</i>	if an error occurs.
<i>IndexOutOfBoundsException</i>	<i>start</i> is greater than <i>end</i> , or <i>end</i> is greater than <i>csq.length()</i>

Implements **decaf::lang::Appendable** (p. 695).

6.876.2.4 virtual **decaf::lang::Appendable&** **decaf::io::Writer::doAppendChar** ( *char value* )  
throw ( **decaf::io::IOException** ) [protected, virtual]

6.876.2.5 virtual **decaf::lang::Appendable&** **decaf::io::Writer::doAppendCharSequence** ( **const decaf::lang::CharSequence \* csq** ) throw ( **decaf::io::IOException** )  
[protected, virtual]

6.876.2.6 virtual **decaf::lang::Appendable&** **decaf::io::Writer::doAppendCharSequenceStartEnd** ( **const decaf::lang::CharSequence \* csq**, *int start*, *int end* ) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::IndexOutOfBoundsException** ) [protected, virtual]

6.876.2.7 virtual void **decaf::io::Writer::doWriteArray** ( **const char \* buffer**, *int size* ) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::NullPointerException** )  
[protected, virtual]

6.876.2.8 virtual void **decaf::io::Writer::doWriteArrayBounded** ( **const char \* buffer**, *int size*, *int offset*, *int length* ) throw ( **decaf::io::IOException**, **decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IndexOutOfBoundsException** )  
[protected, pure virtual]

Override this method to customize the functionality of the method **write**( *char\* buffer*, *int size*, *int offset*, *int length* ).

All subclasses must override this method to provide the basic **Writer** (p. 3951) functionality.

Implemented in **decaf::io::OutputStreamWriter** (p. 2866).

- 6.876.2.9 `virtual void decaf::io::Writer::doWriteChar ( char v ) throw ( decaf::io::IOException )` [protected, virtual]
- 6.876.2.10 `virtual void decaf::io::Writer::doWriteString ( const std::string & str ) throw ( decaf::io::IOException )` [protected, virtual]
- 6.876.2.11 `virtual void decaf::io::Writer::doWriteStringBounded ( const std::string & str, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )` [protected, virtual]
- 6.876.2.12 `virtual void decaf::io::Writer::doWriteVector ( const std::vector< char > & buffer ) throw ( decaf::io::IOException )` [protected, virtual]
- 6.876.2.13 `virtual void decaf::io::Writer::write ( char v ) throw ( decaf::io::IOException )` [virtual]

Writes an single byte char value.

#### Parameters

<i>v</i>	The value to be written.
----------	--------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	thrown if an error occurs.
--	----------------------------

- 6.876.2.14 `virtual void decaf::io::Writer::write ( const std::string & str ) throw ( decaf::io::IOException )` [virtual]

Writes a string.

#### Parameters

<i>str</i>	The string to be written.
------------	---------------------------

#### Exceptions

<b><i>IOException</i></b> (p. 2103)	thrown if an error occurs.
--	----------------------------

- 6.876.2.15 `virtual void decaf::io::Writer::write ( const std::string & str, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Writes a string.

**Parameters**

<i>str</i>	The string to be written.
<i>offset</i>	The position in the array to start writing from.
<i>length</i>	The number of bytes in the array to write.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	thrown if an error occurs.
<i>IndexOutOfBoundsException</i>	if offset+length is greater than the string length.

6.876.2.16 `virtual void decaf::io::Writer::write ( const char * buffer, int size ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException ) [virtual]`

Writes a byte array to the output stream.

**Parameters**

<i>buffer</i>	The byte array to write (cannot be NULL).
<i>size</i>	The size in bytes of the buffer passed.

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	if an I/O error occurs.
<i>NullPointerException</i>	if <i>buffer</i> is NULL.

6.876.2.17 `virtual void decaf::io::Writer::write ( const std::vector< char > & buffer ) throw ( decaf::io::IOException ) [virtual]`

Writes an array of Chars.

**Parameters**

<i>buffer</i>	The array to be written.
---------------	--------------------------

**Exceptions**

<b><i>IOException</i></b> (p. 2103)	thrown if an error occurs.
--	----------------------------

6.876.2.18 `virtual void decaf::io::Writer::write ( const char * buffer, int size, int offset, int length ) throw ( decaf::io::IOException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IndexOutOfBoundsException )` [virtual]

Writes a byte array to the output stream.

#### Parameters

<i>buffer</i>	The byte array to write (cannot be NULL).
<i>size</i>	The size in bytes of the buffer passed.
<i>offset</i>	The position in the array to start writing from.
<i>length</i>	The number of bytes in the array to write.

#### Exceptions

<b>IOException</b> (p. 2103)	if an I/O error occurs.
<i>NullPointerException</i>	if buffer is NULL.
<i>IndexOutOfBoundsException</i>	if offset + length > size of the buffer.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Writer.h`

## 6.877 decaf::security::auth::x500::X500Principal Class Reference

```
#include <src/main/decaf/security/auth/x500/X500Principal.h>
```

Inheritance diagram for decaf::security::auth::x500::X500Principal:

#### Public Member Functions

- virtual `~X500Principal ()`
- virtual `std::string getName () const =0`  
*Provides the name of this principal.*
- virtual void `getEncoded (std::vector< unsigned char > &output) const =0`
- virtual `int hashCode () const =0`

#### 6.877.1 Constructor & Destructor Documentation

6.877.1.1 `virtual decaf::security::auth::x500::X500Principal::X500Principal ( )` [inline, virtual]



## 6.877.2 Member Function Documentation

6.877.2.1 virtual void decaf::security::auth::x500::X500Principal::getEncoded ( std::vector< unsigned char > & *output* ) const [pure virtual]

6.877.2.2 virtual std::string decaf::security::auth::x500::X500Principal::getName ( ) const [pure virtual]

Provides the name of this principal.

### Returns

the name of this principal.

Implements **decaf::security::Principal** (p. 2975).

6.877.2.3 virtual int decaf::security::auth::x500::X500Principal::hashCode ( ) const [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/auth/x500/**X500Principal.h**

## 6.878 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/X509Certificate.h>
```

Inheritance diagram for decaf::security::cert::X509Certificate:

### Public Member Functions

- virtual ~**X509Certificate** ()
- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const decaf::util::Date &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0
- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal \* **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0
- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0
- virtual std::string **getSigAlgName** () const =0

- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal \* **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0  
throw ( CertificateEncodingException )
- virtual int **getVersion** () const =0

### 6.878.1 Detailed Description

Base interface for all identity certificates.

### 6.878.2 Constructor & Destructor Documentation

- 6.878.2.1 virtual decaf::security::cert::X509Certificate::~X509Certificate ( ) [inline, virtual]

### 6.878.3 Member Function Documentation

- 6.878.3.1 virtual void decaf::security::cert::X509Certificate::checkValidity ( ) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]
- 6.878.3.2 virtual void decaf::security::cert::X509Certificate::checkValidity ( const decaf::util::Date & date ) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]
- 6.878.3.3 virtual int decaf::security::cert::X509Certificate::getBasicConstraints ( ) const [pure virtual]
- 6.878.3.4 virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID ( std::vector< bool > & output ) const [pure virtual]
- 6.878.3.5 virtual const X500Principal\* decaf::security::cert::X509Certificate::getIssuerX500Principal ( ) const [pure virtual]
- 6.878.3.6 virtual void decaf::security::cert::X509Certificate::getKeyUsage ( std::vector< unsigned char > & output ) const [pure virtual]
- 6.878.3.7 virtual Date decaf::security::cert::X509Certificate::getNotAfter ( ) const [pure virtual]
- 6.878.3.8 virtual Date decaf::security::cert::X509Certificate::getNotBefore ( ) const [pure virtual]

- 6.878.3.9 virtual std::string decaf::security::cert::X509Certificate::getSigAlgName ( ) const  
[pure virtual]
- 6.878.3.10 virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID ( ) const  
[pure virtual]
- 6.878.3.11 virtual void decaf::security::cert::X509Certificate::getSigAlgParams ( std::vector< unsigned char > & output ) const [pure virtual]
- 6.878.3.12 virtual void decaf::security::cert::X509Certificate::getSignature ( std::vector< unsigned char > & output ) const [pure virtual]
- 6.878.3.13 virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID ( std::vector< bool > & output ) const [pure virtual]
- 6.878.3.14 virtual const X500Principal\* decaf::security::cert::X509Certificate::getSubjectX500Principal ( ) const [pure virtual]
- 6.878.3.15 virtual void decaf::security::cert::X509Certificate::getTBSCertificate ( std::vector< unsigned char > & output ) const throw ( CertificateEncodingException )  
[pure virtual]
- 6.878.3.16 virtual int decaf::security::cert::X509Certificate::getVersion ( ) const [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/X509Certificate.h

## 6.879 activemq::commands::XATransactionId Class Reference

```
#include <src/main/activemq/commands/XATransactionId.h>
```

Inheritance diagram for activemq::commands::XATransactionId:

### Public Types

- typedef decaf::lang::PointerComparator< XATransactionId > COMPARATOR

### Public Member Functions

- XATransactionId ( )
- XATransactionId (const XATransactionId &other)
- virtual ~XATransactionId ( )

- virtual unsigned char **getDataStructureType** () const  
*Get the unique identifier that this object and its own Marshaler share.*
- virtual **XATransactionId** \* **cloneDataStructure** () const  
*Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.*
- virtual void **copyDataStructure** (const **DataStructure** \*src)  
*Copy the contents of the passed object into this object's members, overwriting any existing data.*
- virtual std::string **toString** () const  
*Returns a string containing the information for this **DataStructure** (p. 1628) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** \*value) const  
*Compares the **DataStructure** (p. 1628) passed in to this one, and returns if they are equivalent.*
- virtual int **getFormatId** () const
- virtual void **setFormatId** (int formatId)
- virtual const std::vector< unsigned char > & **getGlobalTransactionId** () const
- virtual std::vector< unsigned char > & **getGlobalTransactionId** ()
- virtual void **setGlobalTransactionId** (const std::vector< unsigned char > &globalTransactionId)
- virtual const std::vector< unsigned char > & **getBranchQualifier** () const
- virtual std::vector< unsigned char > & **getBranchQualifier** ()
- virtual void **setBranchQualifier** (const std::vector< unsigned char > &branchQualifier)
- virtual int **compareTo** (const **XATransactionId** &value) const
- virtual bool **equals** (const **XATransactionId** &value) const
- virtual bool **operator==** (const **XATransactionId** &value) const
- virtual bool **operator<** (const **XATransactionId** &value) const
- **XATransactionId** & **operator=** (const **XATransactionId** &other)

### Static Public Attributes

- static const unsigned char **ID\_XATRANSACTIONID** = 112

### Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

## 6.879.1 Member Typedef Documentation

- 6.879.1.1 **typedef decaf::lang::PointerComparator<XATransactionId>  
activemq::commands::XATransactionId::COMPARATOR**

Reimplemented from **activemq::commands::TransactionId** (p. 3760).

## 6.879.2 Constructor & Destructor Documentation

6.879.2.1 `activemq::commands::XATransactionId::XATransactionId ( )`

6.879.2.2 `activemq::commands::XATransactionId::XATransactionId ( const XATransactionId & other )`

6.879.2.3 `virtual activemq::commands::XATransactionId::~~XATransactionId ( )`  
[virtual]

## 6.879.3 Member Function Documentation

6.879.3.1 `virtual XATransactionId* activemq::commands::XATransactionId::cloneDataStructure ( ) const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

### Returns

new copy of this object.

Reimplemented from `activemq::commands::TransactionId` (p. 3761).

6.879.3.2 `virtual int activemq::commands::XATransactionId::compareTo ( const XATransactionId & value ) const` [virtual]

6.879.3.3 `virtual void activemq::commands::XATransactionId::copyDataStructure ( const DataStructure * src )` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

### Parameters

<code>src</code>	- Source Object
------------------	-----------------

Reimplemented from `activemq::commands::TransactionId` (p. 3761).

6.879.3.4 `virtual bool activemq::commands::XATransactionId::equals ( const DataStructure * value ) const` [virtual]

Compares the `DataStructure` (p. 1628) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

**Returns**

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 3761).

- 6.879.3.5 `virtual bool activemq::commands::XATransactionId::equals ( const XATransactionId & value ) const [virtual]`
- 6.879.3.6 `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ( ) const [virtual]`
- 6.879.3.7 `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier ( ) [virtual]`
- 6.879.3.8 `virtual unsigned char activemq::commands::XATransactionId::getDataStructureType ( ) const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

**Returns**

new **DataStructure** (p. 1628) type copy.

Reimplemented from **activemq::commands::TransactionId** (p. 3762).

- 6.879.3.9 `virtual int activemq::commands::XATransactionId::getFormatId ( ) const [virtual]`
- 6.879.3.10 `virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId ( ) const [virtual]`
- 6.879.3.11 `virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId ( ) [virtual]`
- 6.879.3.12 `virtual bool activemq::commands::XATransactionId::operator< ( const XATransactionId & value ) const [virtual]`
- 6.879.3.13 `XATransactionId& activemq::commands::XATransactionId::operator= ( const XATransactionId & other )`
- 6.879.3.14 `virtual bool activemq::commands::XATransactionId::operator== ( const XATransactionId & value ) const [virtual]`

## 6.880 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller

---

### Class Reference 3977

- 6.879.3.15 `virtual void activemq::commands::XATransactionId::setBranchQualifier ( const std::vector< unsigned char > & branchQualifier ) [virtual]`
- 6.879.3.16 `virtual void activemq::commands::XATransactionId::setFormatId ( int formatId ) [virtual]`
- 6.879.3.17 `virtual void activemq::commands::XATransactionId::setGlobalTransactionId ( const std::vector< unsigned char > & globalTransactionId ) [virtual]`
- 6.879.3.18 `virtual std::string activemq::commands::XATransactionId::toString ( ) const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1628) such as its type and value of its elements.

#### Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 3762).

#### 6.879.4 Field Documentation

- 6.879.4.1 `std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier [protected]`
- 6.879.4.2 `int activemq::commands::XATransactionId::formatId [protected]`
- 6.879.4.3 `std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId [protected]`
- 6.879.4.4 `const unsigned char activemq::commands::XATransactionId::ID_ - XATRANSACTIONID = 112 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/XATransactionId.h`

## 6.880 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller

---

### Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3964).

```
#include <src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller`:

## Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut, **utils::BooleanStream** \*bs) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataInputStream** \*dataIn) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat** \*wireFormat, **commands::DataStructure** \*dataStructure, **decaf::io::DataOutputStream** \*dataOut) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.880.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3964).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

### 6.880.2 Constructor & Destructor Documentation

- 6.880.2.1 `activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::XATransactionIdMarshaller`  
( ) [inline]



## 6.880 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller

### Class Reference 3979

6.880.2.2 virtual activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::~XATransactionIdMarshaller ( ) [inline, virtual]

### 6.880.3 Member Function Documentation

6.880.3.1 virtual commands::DataStructure\* activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::createObject ( ) const [virtual]

Creates a new instance of this marshalable type.

#### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.880.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]

Get the Data Structure Type that identifies this Marshaler.

#### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.880.3.3 virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat \* wireFormat, commands::DataStructure \* dataStructure, decaf::io::DataOutputStream \* dataOut ) throw ( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3782).

```

6.880.3.4  virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]

```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3783).

```

6.880.3.5  virtual int activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]

```

Write the booleans that this object uses to a BooleanStream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Returns

int value indicating the size of the marshaled object.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller** (p. 3783).

## 6.880 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller

### Class Reference 3981

6.880.3.6 virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightMarshal2  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \* *dataStructure*,  
decaf::io::DataOutputStream \* *dataOut*, utils::BooleanStream \* *bs* ) throw  
( decaf::io::IOException ) [virtual]

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**  
(p. 3784).

6.880.3.7 virtual void activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller::tightUnmarshal  
( OpenWireFormat \* *wireFormat*, commands::DataStructure \*  
*dataStructure*, decaf::io::DataInputStream \* *dataIn*, utils::BooleanStream \*  
*bs* ) throw ( decaf::io::IOException ) [virtual]

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**  
(p. 3784).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h

## 6.881 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3968).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/XATransactionId
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.881.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3968).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.881.2 Constructor & Destructor Documentation

6.881.2.1 `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::XATransactionIdMarshaller ( ) [inline]`

6.881.2.2 `virtual activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::~~XATransactionIdMarshaller ( ) [inline, virtual]`

## 6.881.3 Member Function Documentation

6.881.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.881.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.881.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3771).

```
6.881.3.4  virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3772).

```
6.881.3.5  virtual int activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.881 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller

---

Class Reference3985

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3772).

6.881.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3773).

6.881.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3773).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**XATransactionIdMarshaller.h**

## 6.882 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3972).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/XATransactionId
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.882.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3972).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.882.2 Constructor & Destructor Documentation

6.882.2.1 `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::XATransactionIdMarshaller ( ) [inline]`

6.882.2.2 `virtual activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::~~XATransactionIdMarshaller ( ) [inline, virtual]`

## 6.882.3 Member Function Documentation

6.882.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.882.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.882.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3779).

```
6.882.3.4  virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3779).

```
6.882.3.5  virtual int activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.882 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller

### Class Reference 3989

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3780).

```
6.882.3.6 virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3780).

```
6.882.3.7 virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3781).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**XATransactionIdMarshaller.h**

## 6.883 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3976).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/XATransactionId
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.883.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3976).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.883.2 Constructor & Destructor Documentation

6.883.2.1 `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::XATransactionIdMarshaller ( ) [inline]`

6.883.2.2 `virtual activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::~~XATransactionIdMarshaller ( ) [inline, virtual]`

## 6.883.3 Member Function Documentation

6.883.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.883.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.883.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3767).

```
6.883.3.4  virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3768).

```
6.883.3.5  virtual int activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.883 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller

### Class Reference 3993

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3768).

```
6.883.3.6 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3769).

```
6.883.3.7 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3769).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**XATransactionIdMarshaller.h**

## 6.884 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3980).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/XATransactionId
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.884.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3980).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module



## 6.884.2 Constructor & Destructor Documentation

6.884.2.1 `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::XATransactionIdMarshaller ( ) [inline]`

6.884.2.2 `virtual activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::~~XATransactionIdMarshaller ( ) [inline, virtual]`

## 6.884.3 Member Function Documentation

6.884.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.884.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.884.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3775).

```
6.884.3.4  virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3775).

```
6.884.3.5  virtual int activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.884 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller

### Class Reference 3997

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3776).

6.884.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal2 ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3776).

6.884.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightUnmarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]`

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3777).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**XATransactionIdMarshaller.h**

## 6.885 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller

### Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3984).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/XATransactionId
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller:

### Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure \* createObject** () const  
*Creates a new instance of this marshalable type.*
- virtual unsigned char **getDataStructureType** () const  
*Get the Data Structure Type that identifies this Marshaler.*
- virtual void **tightUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual int **tightMarshal1** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write the booleans that this object uses to a BooleanStream.*
- virtual void **tightMarshal2** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**, **utils::BooleanStream \*bs**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*
- virtual void **looseUnmarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataInputStream \*dataIn**) throw ( decaf::io::IOException )  
*Un-marshal an object instance from the data input stream.*
- virtual void **looseMarshal** (**OpenWireFormat \*wireFormat**, **commands::DataStructure \*dataStructure**, **decaf::io::DataOutputStream \*dataOut**) throw ( decaf::io::IOException )  
*Write a object instance to data output stream.*

### 6.885.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3984).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

## 6.885.2 Constructor & Destructor Documentation

6.885.2.1 `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::XATransactionIdMarshaller ( ) [inline]`

6.885.2.2 `virtual activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::~~XATransactionIdMarshaller ( ) [inline, virtual]`

## 6.885.3 Member Function Documentation

6.885.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::createObject ( ) const [virtual]`

Creates a new instance of this marshalable type.

### Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1578).

6.885.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::getDataStructureType ( ) const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

### Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1585).

6.885.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseMarshal ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw ( decaf::io::IOException ) [virtual]`

Write a object instance to data output stream.

### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryWriter that provides that data sink

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3764).

```
6.885.3.4  virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataInputStream * dataIn ) throw ( decaf::io::IOException )
[virtual]
```

Un-marshal an object instance from the data input stream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataIn</i>	- BinaryReader that provides that data source

**Exceptions**

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3764).

```
6.885.3.5  virtual int activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
[virtual]
```

Write the booleans that this object uses to a BooleanStream.

**Parameters**

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

**Returns**

int value indicating the size of the marshaled object.

**Exceptions**

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

## 6.885 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller

### Class Reference 4001

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3765).

```
6.885.3.6 virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure,
  decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs ) throw
( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker
<i>dataStructure</i>	- Object to be marshaled
<i>dataOut</i>	- BinaryReader that provides that data sink
<i>bs</i>	- BooleanStream stream used to pack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the marshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3765).

```
6.885.3.7 virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
  dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream *
  bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

#### Parameters

<i>wireFormat</i>	- describes the wire format of the broker.
<i>dataStructure</i>	- Object to be un-marshaled.
<i>dataIn</i>	- BinaryReader that provides that data.
<i>bs</i>	- BooleanStream stream used to unpack bits from the wire.

#### Exceptions

<i>IOException</i>	if an error occurs during the unmarshal.
--------------------	--

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3766).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**XATransactionIdMarshaller.h**

## 6.886 decaf::util::logging::XMLFormatter Class Reference

Format a **LogRecord** (p. 2370) into a standard XML format.

```
#include <src/main/decaf/util/logging/XMLFormatter.h>
```

Inheritance diagram for decaf::util::logging::XMLFormatter:

### Public Member Functions

- **XMLFormatter** ()
- virtual **~XMLFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const  
*Converts a **LogRecord** (p. 2370) into an XML string.*
- virtual std::string **getHead** (const **Handler** \*handler)  
*Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.*
- virtual std::string **getTail** (const **Handler** \*handler)  
*Returns the tail string for a set of log records formatted as XML strings.*

### 6.886.1 Detailed Description

Format a **LogRecord** (p. 2370) into a standard XML format.

TODO - Currently only outputs UTF-8 The **XMLFormatter** (p. 3988) can be used with arbitrary character encodings, but it is recommended that it normally be used with UTF-8. The character encoding can be set on the output **Handler** (p. 1941).

#### Since

1.0

### 6.886.2 Constructor & Destructor Documentation

6.886.2.1 decaf::util::logging::XMLFormatter::XMLFormatter ( )

6.886.2.2 virtual decaf::util::logging::XMLFormatter::~XMLFormatter ( ) [virtual]

### 6.886.3 Member Function Documentation

6.886.3.1 virtual std::string decaf::util::logging::XMLFormatter::format ( const **LogRecord** &record ) const [virtual]

Converts a **LogRecord** (p. 2370) into an XML string.



**Parameters**

<i>record</i>	The log record to be formatted.
---------------	---------------------------------

**Returns**

the log record formatted as an XML string.

Implements **decaf::util::logging::Formatter** (p. 1928).

**6.886.3.2** `virtual std::string decaf::util::logging::XMLFormatter::getHead ( const Handler * handler ) [virtual]`

Returns the header string for a set of log records formatted as XML strings, using the output handler's encoding if it is defined, otherwise using the default platform encoding.

**Parameters**

<i>handler</i>	The output handler, may be NULL.
----------------	----------------------------------

**Returns**

the header string for log records formatted as XML strings.

**6.886.3.3** `virtual std::string decaf::util::logging::XMLFormatter::getTail ( const Handler * handler ) [virtual]`

Returns the tail string for a set of log records formatted as XML strings.

**Parameters**

<i>handler</i>	The output handler, may be NULL.
----------------	----------------------------------

**Returns**

the tail string for log records formatted as XML strings.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/XMLFormatter.h`

## 6.887 `z_stream_s` Struct Reference

```
#include <src/main/decaf/internal/util/zip/zlib.h>
```

**Data Fields**

- **Bytef \* next\_in**

- **uint avail\_in**
- **uLong total\_in**
- **Bytef \* next\_out**
- **uint avail\_out**
- **uLong total\_out**
- **char \* msg**
- **struct internal\_state FAR \* state**
- **alloc\_func zalloc**
- **free\_func zfree**
- **voidpf opaque**
- **int data\_type**
- **uLong Adler**
- **uLong reserved**

#### 6.887.1 Field Documentation

- 6.887.1.1 **uLong z\_stream\_s::Adler**
- 6.887.1.2 **uint z\_stream\_s::avail\_in**
- 6.887.1.3 **uint z\_stream\_s::avail\_out**
- 6.887.1.4 **int z\_stream\_s::data\_type**
- 6.887.1.5 **char\* z\_stream\_s::msg**
- 6.887.1.6 **Bytef\* z\_stream\_s::next\_in**
- 6.887.1.7 **Bytef\* z\_stream\_s::next\_out**
- 6.887.1.8 **voidpf z\_stream\_s::opaque**
- 6.887.1.9 **uLong z\_stream\_s::reserved**
- 6.887.1.10 **struct internal\_state FAR\* z\_stream\_s::state**
- 6.887.1.11 **uLong z\_stream\_s::total\_in**
- 6.887.1.12 **uLong z\_stream\_s::total\_out**
- 6.887.1.13 **alloc\_func z\_stream\_s::zalloc**
- 6.887.1.14 **free\_func z\_stream\_s::zfree**

The documentation for this struct was generated from the following file:

- **src/main/decaf/internal/util/zip/zlib.h**

## 6.888 decaf::util::zip::ZipException Class Reference

```
#include <src/main/decaf/util/zip/ZipException.h>
```

Inheritance diagram for decaf::util::zip::ZipException:

### Public Member Functions

- **ZipException** () throw ()  
*Default Constructor.*
- **ZipException** (const lang::Exception &ex) throw ()  
*Copy Constructor.*
- **ZipException** (const ZipException &ex) throw ()  
*Copy Constructor.*
- **ZipException** (const char \*file, const int lineNumber, const std::exception \*cause, const char \*msg,...) throw ()  
*Constructor - Initializes the file name and line number where this message occurred.*
- **ZipException** (const std::exception \*cause) throw ()  
*Constructor.*
- **ZipException** (const char \*file, const int lineNumber, const char \*msg,...) throw ()  
*Constructor.*
- virtual **ZipException** \* clone () const  
*Clones this exception.*
- virtual ~**ZipException** () throw ()

### 6.888.1 Constructor & Destructor Documentation

6.888.1.1 decaf::util::zip::ZipException::ZipException ( ) throw () [inline]

Default Constructor.

6.888.1.2 decaf::util::zip::ZipException::ZipException ( const lang::Exception & ex ) throw () [inline]

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy
-----------	-----------------------

6.888.1.3 `decaf::util::zip::ZipException::ZipException ( const ZipException & ex ) throw ()`  
`[inline]`

Copy Constructor.

#### Parameters

<i>ex</i>	the exception to copy, which is an instance of this type
-----------	--

6.888.1.4 `decaf::util::zip::ZipException::ZipException ( const char * file, const int lineNumber, const std::exception * cause, const char * msg, ... ) throw ()` `[inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>cause</i>	The exception that was the cause for this one to be thrown.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.888.1.5 `decaf::util::zip::ZipException::ZipException ( const std::exception * cause ) throw ()`  
`[inline]`

Constructor.

#### Parameters

<i>cause</i>	Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.
--------------	--

6.888.1.6 `decaf::util::zip::ZipException::ZipException ( const char * file, const int lineNumber, const char * msg, ... ) throw ()` `[inline]`

Constructor.

#### Parameters

<i>file</i>	The file name where exception occurs
<i>lineNumber</i>	The line number where the exception occurred.
<i>msg</i>	The message to report
<i>...</i>	list of primitives that are formatted into the message

6.888.1.7 `virtual decaf::util::zip::ZipException::~~ZipException ( ) throw () [inline, virtual]`

## 6.888.2 Member Function Documentation

6.888.2.1 `virtual ZipException* decaf::util::zip::ZipException::clone ( ) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

### Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 2105).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/zip/ZipException.h`



## Chapter 7

# File Documentation

### 7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

#### Data Structures

- class **activemq::cmsutil::CachedConsumer**  
*A cached message consumer contained within a pooled session.*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

### 7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

#### Data Structures

- class **activemq::cmsutil::CachedProducer**  
*A cached message producer contained within a pooled session.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h>
#include <activemq/cmsutil/ResourceLifecycleManager.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

## Data Structures

- class **activemq::cmsutil::CmsAccessor**  
*Base class for **activemq.cmsutil.CmsTemplate** (p. 1140) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1294) to operate on.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsAccessor.h>
#include <activemq/cmsutil/DynamicDestinationResolver.h>
```

## Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**  
*Extends the **CmsAccessor** (p. 1123) to add support for resolving destination names.*



## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsDestinationAccessor.h>
#include <activemq/cmsutil/SessionCallback.h>
#include <activemq/cmsutil/ProducerCallback.h>
#include <activemq/cmsutil/SessionPool.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

## Data Structures

- class **activemq::cmsutil::CmsTemplate**  
***CmsTemplate** (p. 1140) simplifies performing synchronous CMS operations.*
- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::DestinationResolver**  
*Resolves a CMS destination name to a *Destination*.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h>
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**  
*Resolves a CMS destination name to a *Destination*.*
- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**  
*Manages maps of names to topics and queues for a single session.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h>
#include <cms/Message.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::MessageCreator**

*Creates the user-defined message to be sent by the **CmsTemplate** (p. 1140).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>
#include <decaf/util/STLMap.h>
#include <activemq/cmsutil/CachedProducer.h>
#include <activemq/cmsutil/CachedConsumer.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::PooledSession**

*A pooled session object that wraps around a delegate session.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h>
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::ProducerCallback**  
*Callback for sending a message to a CMS destination.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

### 7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference

```
#include <cms/Connection.h>
#include <cms/Session.h>
#include <cms/Destination.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
#include <decaf/util/StlList.h>
```

## Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**  
*Manages the lifecycle of a set of CMS resources.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

### 7.12 src/main/decaf/internal/util/ResourceLifecycleManager.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/util/StlSet.h>
#include <decaf/internal/util/Resource.h>
```

### Data Structures

- class **decaf::internal::util::ResourceLifecycleManager**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.13 src/main/activemq/cmsutil/SessionCallback.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::cmsutil::SessionCallback**  
*Callback for executing any number of operations on a provided CMS Session.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::cmsutil**

## 7.14 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h>
#include <decaf/util/concurrent/Mutex.h>
#include <cms/Connection.h>
#include <list>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::cmsutil::SessionPool**

*A pool of CMS sessions from the same connection and with the same acknowledge mode.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::cmsutil**

## 7.15 src/main/activemq/commands/ActiveMQBlobMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/Message.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::commands::ActiveMQBlobMessage**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.16 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <decaf/io/ByteArrayOutputStream.h>
```

## 7.17 src/main/activemq/commands/ActiveMQDestination.h File Reference 4017

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <cms/BytesMessage.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQBytesMessage**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.17 src/main/activemq/commands/ActiveMQDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/ActiveMQProperties.h>
#include <cms/Destination.h>
#include <decaf/lang/Pointer.h>
#include <vector>
#include <string>
#include <map>
```

### Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.18 src/main/activemq/commands/ActiveMQMapMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <cms/MapMessage.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQMapMessage**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.19 src/main/activemq/commands/ActiveMQMessage.h File Reference

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
```

### Data Structures

- class **activemq::commands::ActiveMQMessage**



## 7.20 `src/main/activemq/commands/ActiveMQMessageTemplate.h` File Reference

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.20 `src/main/activemq/commands/ActiveMQMessageTemplate.h` File Reference

```
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/core/ActiveMQConnection.h>
#include <activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <activemq/util/CMSExceptionSupport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
```

### Data Structures

- class **activemq::commands::ActiveMQMessageTemplate**< T >

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.21 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/ObjectMessage.h>
#include <activemq/util/Config.h>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQObjectMessage**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.22 src/main/activemq/commands/ActiveMQQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Queue.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQQueue**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.23 src/main/activemq/commands/ActiveMQStreamMessage.h File Reference 4021

- namespace **activemq::commands**

## 7.23 src/main/activemq/commands/ActiveMQStreamMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessage.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/StreamMessage.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQStreamMessage**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.24 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
```

```
#include <activemq/exceptions/ActiveMQException.h>
#include <cms/Closeable.h>
#include <vector>
#include <string>
```

### Data Structures

- class **activemq::commands::ActiveMQTempDestination**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::commands**

## 7.25 src/main/activemq/commands/ActiveMQTempQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryQueue.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQTempQueue**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.26 src/main/activemq/commands/ActiveMQTempTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryTopic.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQTempTopic**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.27 src/main/activemq/commands/ActiveMQTextMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/TextMessage.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQTextMessage**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.28 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Topic.h>
#include <vector>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::commands::ActiveMQTopic**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.29 src/main/activemq/commands/BaseCommand.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
```

### Data Structures

- class **activemq::commands::BaseCommand**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.30 src/main/activemq/commands/BaseDataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataSet.h>
#include <string>
#include <sstream>
```

### Data Structures

- class **activemq::commands::BaseDataSet**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::commands**

## 7.31 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataSet.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::commands::BooleanExpression**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.32 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::BrokerError**  
*This class represents an Exception sent from the Broker.*
- struct **activemq::commands::BrokerError::StackTraceElement**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.33 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::BrokerId**

### Namespaces

- namespace **activemq**



*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.34 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::BrokerInfo**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.35 src/main/activemq/commands/Command.h File Reference

```
#include <string>
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::commands::Command**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**
- namespace **activemq::commands**

## 7.36 src/main/activemq/commands/ConnectionControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ConnectionControl**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.37 src/main/activemq/commands/ConnectionError.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ConnectionError**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.38 src/main/activemq/commands/ConnectionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ConnectionId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.39 src/main/activemq/commands/ConnectionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

```
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConnectionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.40 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConsumerControl**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.41 src/main/activemq/commands/ConsumerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
```

```
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConsumerId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.42 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BooleanExpression.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ConsumerInfo**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

### 7.43 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

#### Data Structures

- class **activemq::commands::ControlCommand**

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

### 7.44 src/main/activemq/commands/DataArrayResponse.h File Reference

```
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

#### Data Structures

- class **activemq::commands::DataArrayResponse**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.45 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::DataResponse**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.46 src/main/activemq/commands/DataSet.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

## Data Structures

- class **activemq::commands::DataSet**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.47 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::DestinationInfo**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.48 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::DiscoveryEvent**



## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.49 src/main/activemq/commands/ExceptionResponse.h File Reference

```
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ExceptionResponse**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.50 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::FlushCommand**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.51 src/main/activemq/commands/IntegerResponse.h File Reference

```
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::IntegerResponse**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::commands**

## 7.52 src/main/activemq/commands/JournalQueueAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::JournalQueueAck**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.53 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::JournalTopicAck**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.54 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
```

```
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalTrace**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.55 src/main/activemq/commands/JournalTransaction.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::JournalTransaction**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.56 src/main/activemq/commands/KeepAliveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

## 7.57 src/main/activemq/commands/LastPartialCommand.h File Reference 4039

```
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::KeepAliveInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.57 src/main/activemq/commands/LastPartialCommand.h File Reference

```
#include <activemq/commands/PartialCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::LastPartialCommand**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.58 src/main/activemq/commands/LocalTransactionId.h File Reference

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
```

```
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::LocalTransactionId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.59 src/main/activemq/commands/Message.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::Message**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::commands**

## 7.60 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <cms/CMSException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

## Data Structures

- class **cms::Message**  
*Root of all messages.*

## Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.61 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

```
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::MessageAck**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.62 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::MessageDispatch**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**



## 7.63 src/main/activemq/commands/MessageDispatchNotification.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::MessageDispatchNotification**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.64 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::MessageId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

### 7.65 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::MessagePull**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

### 7.66 src/main/activemq/commands/NetworkBridgeFilter.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::NetworkBridgeFilter**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.67 src/main/activemq/commands/PartialCommand.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::PartialCommand**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.68 src/main/activemq/commands/ProducerAck.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ProducerAck**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.69 src/main/activemq/commands/ProducerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ProducerId**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.70 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/RemoveInfo.h>
```

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::ProducerInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.71 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

## Data Structures

- class **activemq::commands::RemoveInfo**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.72 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.73 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ReplayCommand**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.74 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::Response**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.75 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::SessionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.76 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::SessionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.77 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::ShutdownInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**



## 7.78 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::SubscriptionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.79 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::TransactionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.80 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::TransactionInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.81 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <vector>
```

### Data Structures

- class **activemq::commands::WireFormatInfo**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.82 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

### Data Structures

- class **activemq::commands::XATransactionId**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**

## 7.83 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::core::ActiveMQAckHandler**  
*Interface class that is used to give CMS Messages an interface to Ack themselves with.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**
- namespace **activemq::core**

## 7.84 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::core::ActiveMQConnection**  
*Concrete connection used for all connectors to the ActiveMQ broker.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.85 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
```

## 7.86 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference 4055

```
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
```

### Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.86 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionMetaData.h>
```

### Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**  
*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 244) class.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string>
#include <map>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::core::ActiveMQConstants**  
*Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.*
- class **activemq::core::ActiveMQConstants::StaticInitializer**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/core/RedeliveryPolicy.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

## Data Structures

- class **activemq::core::ActiveMQConsumer**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.89 src/main/activemq/core/ActiveMQProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/util/MemoryUsage.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <memory>
```

## Data Structures

- class **activemq::core::ActiveMQProducer**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.90 src/main/activemq/core/ActiveMQQueueBrowser.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/MessageEnumeration.h>
```

```
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

## Data Structures

- class **activemq::core::ActiveMQQueueBrowser**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.91 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h>
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/Pointer.h>
```



```
#include <decaf/util/StlMap.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/Properties.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::core::ActiveMQSession**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.92 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/threads/Task.h>
#include <activemq/threads/TaskRunner.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::core::ActiveMQSessionExecutor**  
*Delegate dispatcher for a single session.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

### 7.93 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/core/Synchronization.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
```

#### Data Structures

- class **activemq::core::ActiveMQTransactionContext**

*Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.*

#### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

### 7.94 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h>
#include <memory>
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::core::DispatchData**

*Simple POCO that contains the information necessary to route a message to a specified consumer.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.95 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::core::Dispatcher**

*Interface for an object responsible for dispatching messages to consumers.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.96 src/main/activemq/core/MessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/MessageDispatch.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::core::MessageDispatchChannel**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.97 src/main/activemq/core/policies/DefaultPrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/PrefetchPolicy.h>
```

## Data Structures

- class **activemq::core::policies::DefaultPrefetchPolicy**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::core::policies**

## 7.98 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/RedeliveryPolicy.h>
```

## Data Structures

- class **activemq::core::policies::DefaultRedeliveryPolicy**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**
- namespace **activemq::core::policies**

## 7.99 src/main/activemq/core/PrefetchPolicy.h File Reference

```
#include <activemq/util/Config.h>  
#include <decaf/util/Properties.h>
```

## Data Structures

- class **activemq::core::PrefetchPolicy**  
*Interface for a Policy object that controls message Prefetching on various destination types in ActiveMQ-CPP.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::core**

## 7.100 src/main/activemq/core/RedeliveryPolicy.h File Reference

```
#include <activemq/util/Config.h>  
#include <decaf/util/Properties.h>
```

## Data Structures

- class **activemq::core::RedeliveryPolicy**  
*Interface for a **RedeliveryPolicy** (p. 3121) object that controls how message Redelivery is handled in ActiveMQ-CPP when a transaction is rolled back.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.101 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
```

### Data Structures

- class **activemq::core::Synchronization**

*Transacted Object **Synchronization** (p. 3659), used to sync the events of a Transaction with the items in the Transaction.*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::core**

## 7.102 src/main/activemq/exceptions/ActiveMQException.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <decaf/lang/Exception.h>
#include <activemq/exceptions/ExceptionDefines.h>
#include <stdarg.h>
#include <sstream>
```

### Data Structures

- class **activemq::exceptions::ActiveMQException**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::exceptions**

## 7.103 src/main/activemq/exceptions/BrokerException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/BrokerError.h>
#include <sstream>
```

### Data Structures

- class **activemq::exceptions::BrokerException**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::exceptions**

## 7.104 src/main/activemq/exceptions/ExceptionDefines.h File Reference

### Defines

- #define **AMQ\_CATCH\_RETHROW**(type)  
*Macro for catching and re-throwing an exception of a given type.*
- #define **AMQ\_CATCH\_EXCEPTION\_CONVERT**(sourceType, targetType)  
*Macro for catching an exception of one type and then re-throwing as another type.*
- #define **AMQ\_CATCHALL\_THROW**(type)  
*A catch-all that throws a known exception.*
- #define **AMQ\_CATCHALL\_NOTHROW**()  
*A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.*
- #define **AMQ\_CATCH\_NOTHROW**(type)  
*Macro for catching and re-throwing an exception of a given type.*

### 7.104.1 Define Documentation

#### 7.104.1.1 `#define AMQ_CATCH_EXCEPTION_CONVERT( sourceType, targetType )`

##### Value:

```
catch( sourceType& ex ){ \
    targetType target( ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then re-throwing as another type.

##### Parameters

<i>sourceType</i>	the type of the exception to be caught.
<i>targetType</i>	the type of the exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

#### 7.104.1.2 `#define AMQ_CATCH_NOTHROW( type )`

##### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and re-throwing an exception of a given type.

##### Parameters

<i>type</i>	The type of the exception to throw (e.g. <code>ActiveMQException</code> ).
-------------	--

#### 7.104.1.3 `#define AMQ_CATCH_RETHROW( type )`

##### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and re-throwing an exception of a given type.

##### Parameters

<i>type</i>	The type of the exception to throw (e.g. <code>ActiveMQException</code> ).
-------------	--



Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

#### 7.104.1.4 #define AMQ\_CATCHALL\_NOTHROW( )

##### Value:

```
catch( ... ){ \
    activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

#### 7.104.1.5 #define AMQ\_CATCHALL\_THROW( type )

##### Value:

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

##### Parameters

<i>type</i>	the type of exception to be thrown.
-------------	-------------------------------------

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

## 7.105 src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference

### Defines

- #define **DECAF\_CATCH\_RETHROW**(type)  
*Macro for catching and rethrowing an exception of a given type.*
- #define **DECAF\_CATCH\_EXCEPTION\_CONVERT**(sourceType, targetType)  
*Macro for catching an exception of one type and then rethrowing as another type.*
- #define **DECAF\_CATCHALL\_THROW**(type)  
*A catch-all that throws a known exception.*

- **#define DECAF\_CATCHALL\_NOTHROW()**  
*A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.*
- **#define DECAF\_CATCH\_NOTHROW(type)**  
*Macro for catching and rethrowing an exception of a given type.*

### 7.105.1 Define Documentation

#### 7.105.1.1 #define DECAF\_CATCH\_EXCEPTION\_CONVERT( sourceType, targetType )

##### Value:

```
catch( sourceType& ex ){ \
    targetType target( &ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then rethrowing as another type.

##### Parameters

<i>sourceType</i>	the type of the exception to be caught.
<i>targetType</i>	the type of the exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`.

#### 7.105.1.2 #define DECAF\_CATCH\_NOTHROW( type )

##### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and rethrowing an exception of a given type.

##### Parameters

<i>type</i>	The type of the exception to throw (e.g. Exception ).
-------------	---

#### 7.105.1.3 #define DECAF\_CATCH\_RETHROW( type )

##### Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and rethrowing an exception of a given type.

#### Parameters

<i>type</i>	The type of the exception to throw (e.g. <code>Exception</code> ).
-------------	--

Referenced by `decaf::util::PriorityQueue< E >::add()`.

##### 7.105.1.4 `#define DECAF_CATCHALL_NOTHROW( )`

#### Value:

```
catch( ... ){ \
    lang::Exception ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

##### 7.105.1.5 `#define DECAF_CATCHALL_THROW( type )`

#### Value:

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

#### Parameters

<i>type</i>	the type of exception to be thrown.
-------------	-------------------------------------

Referenced by `decaf::util::PriorityQueue< E >::add()`.

## 7.106 src/main/activemq/io/LoggingInputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

## Data Structures

- class **activemq::io::LoggingInputStream**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::io**

## 7.107 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
```

## Data Structures

- class **activemq::io::LoggingOutputStream**  
*OutputStream filter that just logs the data being written.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::io**

## 7.108 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::library::ActiveMQCPP**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::library**

## 7.109 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::state::CommandVisitor**  
*Interface for an Object that can visit the various Command Objects that are sent from and to this client.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::commands**
- namespace **activemq::state**

## 7.110 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/state/CommandVisitor.h>
#include <activemq/core/ActiveMQConstants.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/SessionId.h>
```

```
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/RemoveSubscriptionInfo.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessagePull.h>
#include <activemq/commands/TransactionInfo.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/commands/MessageDispatchNotification.h>
#include <activemq/commands/ControlCommand.h>
#include <activemq/commands/ConnectionError.h>
#include <activemq/commands/ConnectionControl.h>
#include <activemq/commands/ConsumerControl.h>
#include <activemq/commands/ShutdownInfo.h>
#include <activemq/commands/KeepAliveInfo.h>
#include <activemq/commands/FlushCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/ReplayCommand.h>
#include <activemq/commands/Response.h>
```

## Data Structures

- class **activemq::state::CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 1171) that returns NULL for all calls.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::state**

## 7.111 src/main/activemq/state/ConnectionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/STLMap.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentSTLMap.h>
#include <decaf/util/STLList.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

### Data Structures

- class **activemq::state::ConnectionState**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.112 src/main/activemq/state/ConnectionStateTracker.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/state/CommandVisitorAdapter.h>
#include <activemq/state/ConnectionState.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <activemq/state/Tracked.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::state::ConnectionStateTracker**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::state**

## 7.113 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::state::ConsumerState**



## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.114 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ProducerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::state::ProducerState**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.115 src/main/activemq/state/SessionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::state::SessionState**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.116 src/main/activemq/state/Tracked.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::state::Tracked**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.117 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
```

```
#include <string>
#include <memory>
```

## Data Structures

- class **activemq::state::TransactionState**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::state**

## 7.118 src/main/activemq/threads/CompositeTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

## Data Structures

- class **activemq::threads::CompositeTask**  
*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 1194).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::threads**

## 7.119 src/main/activemq/threads/CompositeTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTask.h>
#include <decaf/util/StlSet.h>
```

```
#include <decaf/util/StlList.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::threads::CompositeTaskRunner**

*A **Task** (p. 3678) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::threads**

## 7.120 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/Task.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::threads::DedicatedTaskRunner**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::threads**

## 7.121 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::threads::Task**  
*Represents a unit of work that requires one or more iterations to complete.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::threads**

## 7.122 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h>  
#include <activemq/threads/Task.h>
```

### Data Structures

- class **activemq::threads::TaskRunner**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::threads**

## 7.123 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportFactory.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

## Data Structures

- class **activemq::transport::AbstractTransportFactory**

*Abstract implementation of the **TransportFactory** (p. 3825) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3825) instances.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.124 src/main/activemq/transport/CompositeTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/List.h>
```

## Data Structures

- class **activemq::transport::CompositeTransport**

*A Composite **Transport** (p. 3819) is a **Transport** (p. 3819) implementation that is composed of several **Transports**.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.125 src/main/activemq/transport/correlator/FutureResponse.h File Reference 481

### 7.125 src/main/activemq/transport/correlator/FutureResponse.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
```

#### Data Structures

- class **activemq::transport::correlator::FutureResponse**  
*A container that holds a response object.*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

### 7.126 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/transport/correlator/FutureResponse.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <map>
#include <stdio.h>
```

## Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

*This type of transport filter is responsible for correlating asynchronous responses with requests.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

## 7.127 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::transport::DefaultTransportListener**

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.128 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/DefaultTransportListener.h>
```



```
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

## Data Structures

- class **activemq::transport::failover::BackupTransport**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.129 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/BackupTransport.h>
#include <activemq/transport/failover/URIPool.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/IOException.h>
#include <decaf/util/StlList.h>
```

## Data Structures

- class **activemq::transport::failover::BackupTransportPool**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.130 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::transport::failover::CloseTransportsTask**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.131 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/state/ConnectionStateTracker.h>
#include <activemq/transport/CompositeTransport.h>
#include <activemq/transport/failover/BackupTransportPool.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/FailoverTransportListener.h>
#include <activemq/transport/failover/URIPool.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/StlList.h>
```

```
#include <decaf/util/STLMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/net/URI.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **activemq::transport::failover::FailoverTransport**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

## 7.132 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/transport/Transport.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

## Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**  
*Creates an instance of a **FailoverTransport** (p. 1835).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

### 7.133 **src/main/activemq/transport/failover/FailoverTransportListener.h** File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
```

#### Data Structures

- class **activemq::transport::failover::FailoverTransportListener**

*Utility class used by the **Transport** (p. 3819) to perform the work of responding to events from the active **Transport** (p. 3819).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

### 7.134 **src/main/activemq/transport/failover/URIPool.h** File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

#### Data Structures

- class **activemq::transport::failover::URIPool**

### 7.135 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference 4087

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

### 7.135 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Timer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
```

#### Data Structures

- class **activemq::transport::inactivity::InactivityMonitor**

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

### 7.136 src/main/activemq/transport/inactivity/ReadChecker.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

## Data Structures

- class **activemq::transport::inactivity::ReadChecker**

*Runnable class that is used by the {.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

## 7.137 src/main/activemq/transport/inactivity/WriteChecker.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

## Data Structures

- class **activemq::transport::inactivity::WriteChecker**

*Runnable class used by the {.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

## 7.138 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
```

### 7.139 src/main/activemq/transport/logging/LoggingTransport.h File Reference 4089

```
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <memory>
```

#### Data Structures

- class **activemq::transport::IOTransport**  
*Implementation of the **Transport** (p. 3819) interface that performs marshaling of commands to IO streams.*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

### 7.139 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

#### Data Structures

- class **activemq::transport::logging::LoggingTransport**  
*A transport filter that logs commands as they are sent/received.*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::logging**

## 7.140 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
```

### Data Structures

- class **activemq::transport::mock::InternalCommandListener**  
*Listens for Commands sent from the **MockTransport** (p. 2724).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.141 src/main/activemq/transport/mock/MockTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/transport/DefaultTransportListener.h>
```



## 7.142 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/mock/InternalCommandListener.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <cms/Message.h>
#include <map>
#include <set>
```

### Data Structures

- class **activemq::transport::mock::MockTransport**

The **MockTransport** (p. 2724) defines a base level **Transport** (p. 3819) class that is intended to be used in place of an a regular protocol **Transport** (p. 3819) such as TCP.

### Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

## 7.142 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
```

### Data Structures

- class **activemq::transport::mock::MockTransportFactory**

Manufactures **MockTransports**, which are objects that read from input streams and write to output streams.

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

### 7.143 src/main/activemq/transport/mock/ResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

## Data Structures

- class **activemq::transport::mock::ResponseBuilder**  
*Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

### 7.144 src/main/activemq/transport/tcp/SslTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransport.h>
```

## Data Structures

- class **activemq::transport::tcp::SslTransport**  
***Transport** (p. 3819) for connecting to a Broker using an SSL Socket.*

## 7.145 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference 4093

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.145 src/main/activemq/transport/tcp/SslTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/tcp/TcpTransportFactory.h>
```

### Data Structures

- class **activemq::transport::tcp::SslTransportFactory**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.146 src/main/activemq/transport/tcp/TcpTransport.h File Reference

```
#include <activemq/io/LoggingInputStream.h>
#include <activemq/io/LoggingOutputStream.h>
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/net/Socket.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <memory>
```

## Data Structures

- class **activemq::transport::tcp::TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 2105).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.147 src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/exceptions/ActiveMQException.h>
```

## Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 3696).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

## 7.148 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <typeinfo>
```

### Data Structures

- class **activemq::transport::Transport**  
*Interface for a transport layer for command objects.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::transport**

## 7.149 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::transport::TransportFactory**  
*Defines the interface for Factories that create Transports or TransportFilters.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.150 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
#include <typeinfo>
```

## Data Structures

- class **activemq::transport::TransportFilter**  
*A filter on the transport layer.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::transport**

## 7.151 src/main/activemq/transport/TransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::transport::TransportListener**

*A listener of asynchronous exceptions from a command transport object.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.152 src/main/activemq/transport/TransportRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/transport/TransportFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **activemq::transport::TransportRegistry**

*Registry of all **Transport** (p. 3819) Factories that are available to the client at runtime.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::transport**

## 7.153 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>
#include <string>
```

```
#include <sstream>
#include <activemq/util/Config.h>
#include <cms/CMSProperties.h>
#include <decaf/util/Properties.h>
```

## Data Structures

- class **activemq::util::ActiveMQProperties**  
*Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 3072) object.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.154 src/main/activemq/util/CMSExceptionSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <cms/CMSSecurityException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/InvalidClientIdException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/InvalidSelectorException.h>
#include <cms/IllegalStateException.h>
#include <cms/UnsupportedOperationException.h>
#include <decaf/lang/Exception.h>
#include <string>
```

## Data Structures

- class **activemq::util::CMSExceptionSupport**



- namespace **activemq**

- namespace **activemq::util**

- #define **AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION()**

#### 7.154.1.1 #define AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION( )

```

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::acknowledge(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::clearBody(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::clearProperties(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::equals(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getBooleanProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getByteProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getCMSMessageID(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getDoubleProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getFloatProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getIntProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getLongProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getPropertyNames(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getShortProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::getStringProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::propertyExists(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setBooleanProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setByteProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setCMSDestination(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setCMSReplyTo(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setDoubleProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setFloatProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setIntProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage
>::setLongProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage

```

>::setShortProperty(), and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage`  
>::setStringProperty().

## 7.155 src/main/activemq/util/CompositeData.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/net/URI.h>
#include <decaf/net/URISyntaxException.h>
```

### Data Structures

- class **activemq::util::CompositeData**  
*Represents a Composite URI.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.156 src/main/activemq/util/Config.h File Reference

### Defines

- #define **AMQCPP\_API**
- #define **HAVE\_UUID\_UUID\_H**
- #define **HAVE\_UUID\_T**
- #define **HAVE\_PTHREAD\_H**

### 7.156.1 Define Documentation

7.156.1.1 #define **AMQCPP\_API**

7.156.1.2 #define **HAVE\_PTHREAD\_H**

7.156.1.3 #define **HAVE\_UUID\_T**

7.156.1.4 `#define HAVE_UUID_UUID_H`

## 7.157 src/main/cms/Config.h File Reference

### Defines

- `#define CMS_API`

#### 7.157.1 Define Documentation

7.157.1.1 `#define CMS_API`

## 7.158 src/main/decaf/util/Config.h File Reference

### Defines

- `#define DECAF_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`
- `#define DECAF_UNUSED`

#### 7.158.1 Define Documentation

7.158.1.1 `#define DECAF_API`

7.158.1.2 `#define DECAF_UNUSED`

7.158.1.3 `#define HAVE_PTHREAD_H`

7.158.1.4 `#define HAVE_UUID_T`

7.158.1.5 `#define HAVE_UUID_UUID_H`

## 7.159 src/main/activemq/util/IdGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <string>
```

### Data Structures

- class `activemq::util::IdGenerator`

- class **activemq::util::IdGenerator::StaticData**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.160 src/main/activemq/util/LongSequenceGenerator.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **activemq::util::LongSequenceGenerator**  
*This class is used to generate a sequence of long long values that are incremented each time a new value is requested.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.161 src/main/activemq/util/MarshallingSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <string>
```

## Data Structures

- class **activemq::util::MarshallingSupport**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.162 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **activemq::util::MemoryUsage**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.163 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <stdio.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

## Data Structures

- class **activemq::util::PrimitiveList**  
*List of primitives.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.164 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string>
#include <vector>
#include <activemq/util/Config.h>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

## Data Structures

- class **activemq::util::PrimitiveMap**

*Map of named primitives.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::util**

## 7.165 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <string>
```

## Data Structures

- class **activemq::util::PrimitiveValueConverter**  
*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2960) from one type to another.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.166 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Map.h>
#include <decaf/util/List.h>
```

## Data Structures

- class **activemq::util::PrimitiveValueNode**  
*Class that wraps around a single value of one of the many types.*
- union **activemq::util::PrimitiveValueNode::PrimitiveValue**  
*Define a union type comprised of the various types.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.167 src/main/activemq/util/URISupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/CompositeData.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **activemq::util::URISupport**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.168 src/main/activemq/util/Usage.h File Reference

```
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::util::Usage**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::util**

## 7.169 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>  
#include <decaf/io/IOException.h>  
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::wireformat::MarshalAware**



## 7.170

src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h

File Reference

4107

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

## 7.170 src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
#include <activemq/wireformat/openwire/utils/HexTable.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**  
*Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.171 src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h

### File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::DataStreamMarshaller**

*Base class for all classes that marshal commands for Openwire.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.172 src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/util/PrimitiveList.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/IOException.h>
#include <string>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

*This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.173 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 182).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.174 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 190).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.175 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.176 src/main/activemq/wireformat/openwire/mar-shal/v4/ActiveMQBlobMessageMarshaller.h File

### Reference

4111

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 177).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.176 src/main/activemq/wireformat/openwire/mar-shal/v4/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/mar-shal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 186).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.177 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 194).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.178 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBlobMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
```

## 7.179 src/main/activemq/wireformat/openwire/mar-shal/v1/ActiveMQBytesMessageMarshaller.h File Reference

4113

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 198).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.179 src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 224).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.180 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 240).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**



**7.181 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference**

4115

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

**7.181 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference**

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

**Data Structures**

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 220).*

**Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

**7.182 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference**

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 228).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.183 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 232).*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.184 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 236).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.185 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 308).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.186 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.187 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h File

### Reference

4119

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 320).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.187 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 304).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.188 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 312).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

  - namespace **activemq::wireformat**
  - namespace **activemq::wireformat::openwire**
  - namespace **activemq::wireformat::openwire::marshal**
  - namespace **activemq::wireformat::openwire::marshal::v4**

## 7.189 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

## 7.190 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h File Reference

4121

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 316).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.190 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 324).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.191 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 348).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**



**7.192 src/main/activemq/wireformat/openwire/marsh  
shal/v2/ActiveMQMapMessageMarshaller.h File  
Reference**

**4123**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

**7.192 src/main/activemq/wireformat/openwire/marsh/v2/ActiveMQMapMessageMarshaller.h  
File Reference**

```
#include <activemq/wireformat/openwire/marsh/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

**Data Structures**

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 360).*

**Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

**7.193 src/main/activemq/wireformat/openwire/marsh/v3/ActiveMQMapMessageMarshaller.h  
File Reference**

```
#include <activemq/wireformat/openwire/marsh/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataSeture.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 344).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.194 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSeture.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 352).*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.195 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 356).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.196 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMapMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 364).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.197 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.198 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h File

### Reference

4127

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 375).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.198 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 387).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.199 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 371).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.200 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```

## 7.201 src/main/activemq/wireformat/openwire/mar-shal/v5/ActiveMQMessageMarshaller.h File

### Reference

4129

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 379).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.201 src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 383).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.202 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQMessageMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 391).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**



**7.203 src/main/activemq/wireformat/openwire/marsh  
al/v1/ActiveMQObjectMessageMarshaller.h File  
Reference**

**4131**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

**7.203 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h  
File Reference**

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

**Data Structures**

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 421).*

**Namespaces**

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

**7.204 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h  
File Reference**

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 433).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.205 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 416).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.206 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 425).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.207 `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMar` File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 429).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.208 `src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQObjectMessageMar` File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.209

src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h

### File Reference

4135

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 437).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.209 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 464).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.210 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarsh
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 476).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.211 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarsh
```

## 7.212

**src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h**

### File Reference

4137

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 460).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.212 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 468).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.213 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarsh
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 472).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**



## 7.214

src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h

### File Reference

4139

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.214 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQQueueMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 480).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.215 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 527).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.216 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 539).*

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.217 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 523).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.218 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMar File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 531).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.219 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMar File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.220 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h File Reference

4143

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 535).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.220 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQStreamMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 543).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.221 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 555).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.222 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
```

## 7.223 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

4145

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 566).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.223 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 551).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.224 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 558).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



## 7.225 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

4147

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.225 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 562).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.226 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 570).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.227 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarsha File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationM
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 582).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.228 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 594).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.229 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarsha File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationM
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 578).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.230 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarsha File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationM
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.231 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h File

### Reference

4151

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 586).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.231 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 590).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.232 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempQueueMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 598).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.233 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File Reference

Reference

4153

## 7.233 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec1::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 615).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec1**
- namespace **activemq::wireformat::openwire::marshalsec1::v1**

## 7.234 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 623).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.235 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 607).*

## Namespaces

- namespace **activemq**



## 7.236 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h File Reference

4155

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.236 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 611).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.237 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 619).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.238 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 627).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.239 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 644).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.240 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 656).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.241 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.242 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h File Reference

4159

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 635).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.242 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 640).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.243 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarsh File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 648).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.244 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h File Reference

4161  
7.244 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTextMessageMarshaller.h  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec6::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 652).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec6**
- namespace **activemq::wireformat::openwire::marshalsec6::v6**

## 7.245 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 672).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.246 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 684).*

## Namespaces

- namespace **activemq**



## 7.247

**src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h**

### File Reference

4163

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.247 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 664).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.248 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 668).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.249 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## 7.250

src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h

File Reference

4165

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 676).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.250 src/main/activemq/wireformat/openwire/marshal/v6/ActiveMQTopicMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utls/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 680).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.251 src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 743).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.252 src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.253

src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h

### File Reference

4167

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 764).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.253 src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 730).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.254 src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 737).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.255

src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h

File Reference

7.255 src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h 4169

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 750).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.256 src/main/activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller**

*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 757).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.257 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 840).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*



- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.258 src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 852).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.259 src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 832).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.260 src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 836).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.261 src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 844).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.262 src/main/activemq/wireformat/openwire/marshal/v6/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller**  
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 848).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.263 src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 871).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.264 src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 883).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.265 src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 862).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.266 src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 867).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.267 src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 875).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.268 src/main/activemq/wireformat/openwire/marshal/v6/BrokerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller**  
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 879).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.269 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1250).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.270 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1262).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.271 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1242).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.272 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.273 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h File Reference

4181

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1246).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.273 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1254).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.274 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionControlMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1258).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.275

src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h

File Reference

4183

7.275 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1282).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.276 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1270).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.277 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1274).*

## Namespaces

- namespace **activemq**

7.278

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h

File Reference

4185

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.278 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1278).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.279 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1286).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.280 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionErrorMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



7.281

src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

File Reference

4187

Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1290).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.281 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1313).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.282 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1301).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.283 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.284

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

### File Reference

4189

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1305).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.284 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1309).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.285 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1317).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.286

src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

File Reference

4191

7.286 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionIdMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1321).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.287 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1343).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.288 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1330).*

## Namespaces

- namespace **activemq**

7.289

**src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h**

**File Reference**

**4193**

---

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

**7.289 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h**

**File Reference**

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1335).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

**7.290 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h**

**File Reference**

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1339).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.291 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



## 7.292

**src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h**

**File Reference**

**4195**

**Data Structures**

- class **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1347).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.292 src/main/activemq/wireformat/openwire/marshal/v6/ConnectionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1351).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.293 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1386).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.294 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.295 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h File Reference

4197

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1373).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.295 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1378).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.296 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1382).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.297 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h File Reference

Reference 4199  
7.297 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h  
File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec5::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1390).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec5**
- namespace **activemq::wireformat::openwire::marshalsec5::v5**

## 7.298 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1394).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.299 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1414).*

## Namespaces

- namespace **activemq**

### 7.300

#### src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h File Reference

4201

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.300 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1402).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.301 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1406).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.302 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



### 7.303

#### src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h File Reference

---

4203

##### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1410).*

##### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.303 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

##### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1418).*

##### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.304 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1422).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.305 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.306

src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

#### File Reference

4205

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1447).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.306 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1434).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.307 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1439).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.308

src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h

File Reference

4207

7.308 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1443).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.309 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1451).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.310 src/main/activemq/wireformat/openwire/marshal/v6/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1455).*

## Namespaces

- namespace **activemq**

### 7.311 src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h File Reference

4209

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.311 src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1475).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.312 src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1462).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.313 src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1467).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.314 src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1471).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.315 src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1479).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.316 src/main/activemq/wireformat/openwire/marshal/v6/ControlCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.317 src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h File Reference

4213

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1483).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.317 src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1508).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.318 src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1496).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.319 src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h File Reference

## 7.319 src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h File Reference

4215

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec::v3::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1500).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec**
- namespace **activemq::wireformat::openwire::marshalsec::v3**

### 7.320 src/main/activemq/wireformat/openwire/marshalsec/v4/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshalsec/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1504).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.321 src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1512).*

## Namespaces

- namespace **activemq**

### 7.322 src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h File Reference

4217

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.322 src/main/activemq/wireformat/openwire/marshal/v6/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1516).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.323 src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1573).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.324 src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```



## 7.325

src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h

File Reference

4219

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1561).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.325 src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1565).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.326 src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1569).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.327 src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.328

**src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h**

#### File Reference

4221

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1553).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.328 src/main/activemq/wireformat/openwire/marshal/v6/DataResponseMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1557).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.329 src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1708).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.330

**src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h**

**File Reference**

**7.330** **src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h**

4223

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1696).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.331 **src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h**

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1700).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.332 src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/Util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1704).*

## Namespaces

- namespace **activemq**

### 7.333

**src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h**

#### File Reference

4225

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.333 src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1716).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.334 src/main/activemq/wireformat/openwire/marshal/v6/DestinationInfoMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1712).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.335 src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



7.336

src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h  
File Reference 4227  
Data Structures

---

- class **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1741).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.336 src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1729).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.337 src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1733).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.338 src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.339

**src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h**

#### File Reference

4229

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1737).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.339 src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1745).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.340 src/main/activemq/wireformat/openwire/marshal/v6/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1725).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.341 src/main/activemq/wireformat/openwire/mar-shal/v1/ExceptionResponseMarshaller.h File Reference

Reference

4231

### 7.341 src/main/activemq/wireformat/openwire/mar-shal/v1/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/mar-shal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::mar-shal::v1::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1825).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::mar-shal**
- namespace **activemq::wireformat::openwire::mar-shal::v1**

### 7.342 src/main/activemq/wireformat/openwire/mar-shal/v2/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/mar-shal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1809).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.343 src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1813).*

## Namespaces

- namespace **activemq**

### 7.344 src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h File Reference

4233

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.344 src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1821).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.345 src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1817).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.346 src/main/activemq/wireformat/openwire/marshal/v6/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



7.347

src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h  
File Reference 4235  
Data Structures

---

- class **activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1804).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.347 src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1919).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.348 src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1907).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.349 src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.350

**src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h**  
**File Reference** **4237**

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1911).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.350 src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1915).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.351 src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1923).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.352

src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h

File Reference

7.352 src/main/activemq/wireformat/openwire/marshal/v6/FlushCommandMarshaller.h

4239

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1903).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.353 src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2073).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.354 src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2061).*

## Namespaces

- namespace **activemq**

### 7.355 src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h File Reference

4241

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.355 src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2065).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

### 7.356 src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2069).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.357 src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2077).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.358 src/main/activemq/wireformat/openwire/marshal/v6/IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 2057).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.359 src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2139).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.360 src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.361 src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h File Reference

4245

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2123).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.361 src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2131).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.362 src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller. File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2135).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.363 src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h File Reference

## 7.363 src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h File Reference 4247

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec5::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2127).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec5**
- namespace **activemq::wireformat::openwire::marshalsec5::v5**

### 7.364 src/main/activemq/wireformat/openwire/marshal/v6/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 2119).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.365 src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2168).*

## Namespaces

- namespace **activemq**

### 7.366 src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h File

#### Reference

4249

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.366 src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2152).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

### 7.367 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2156).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.368 src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2164).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.369 src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2148).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.370 src/main/activemq/wireformat/openwire/marshal/v6/JournalTopicAckMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 2160).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.371 src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.372

**src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h**

#### File Reference

4253

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2190).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.372 src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2174).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.373 src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2178).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.374

src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h

File Reference

7.374 src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h

4255

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2186).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.375 src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2194).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.376 src/main/activemq/wireformat/openwire/marshal/v6/JournalTraceMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 2182).*

## Namespaces

- namespace **activemq**

### 7.377 src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h File Reference

4257

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.377 src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2221).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.378 src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2205).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.379 src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2209).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.380 src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2217).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.381 src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2213).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.382 src/main/activemq/wireformat/openwire/marshal/v6/JournalTransactionMarshaller

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.383

src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h

#### File Reference

4261

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 2201).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

### 7.383 src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h

#### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2249).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.384 src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2233).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.385

src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

File Reference

7.385 src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

4263

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2237).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.386 src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2241).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.387 src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2245).*

## Namespaces

- namespace **activemq**

7.388

src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h

File Reference

4265

---

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.388 src/main/activemq/wireformat/openwire/marshal/v6/KeepAliveInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 2228).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.389 src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2283).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.390 src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2271).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.391 src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2267).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.392 src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaler File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2279).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.393 src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaler File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

### 7.394 src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h File Reference

4269

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2275).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.394 src/main/activemq/wireformat/openwire/marshal/v6/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 2262).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.395 src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2330).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

### 7.396 src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h File Reference

## 7.396 src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h File Reference 4271

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec2::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2314).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec**
- namespace **activemq::wireformat::openwire::marshalsec2**

### 7.397 src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2318).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.398 src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2326).*

## Namespaces

- namespace **activemq**

### 7.399 src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h File Reference

4273

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

### 7.399 src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2322).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

### 7.400 src/main/activemq/wireformat/openwire/marshal/v6/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 2310).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.401 src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**



- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.402 src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.403 src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.404 src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.405 src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MarshallerFactory**  
*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.406 src/main/activemq/wireformat/openwire/marshal/v6/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MarshallerFactory**

*Used to create marshallers for a specific version of the wire protocol.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.407 src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>  
#include <decaf/io/DataInputStream.h>  
#include <decaf/io/DataOutputStream.h>  
#include <decaf/io/IOException.h>  
#include <activemq/util/Config.h>  
#include <activemq/commands/DataStructure.h>  
#include <activemq/wireformat/openwire/OpenWireFormat.h>  
#include <activemq/wireformat/openwire/utls/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2542).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.408 src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2530).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.409

src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h

File Reference

7.409 src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h

4279

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2534).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.410 src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2538).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.411 src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2546).*

## Namespaces

- namespace **activemq**

## 7.412

**src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h**

### File Reference

4281

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.412 src/main/activemq/wireformat/openwire/marshal/v6/MessageAckMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2526).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.413 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2582).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.414 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2566).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.415 src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2570).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.416 src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2578).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.417 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.418 src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h File Reference

4285

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2574).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.418 src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2586).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.419 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotification File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2611).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.420 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h File Reference

Reference

4287

## 7.420 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshalsec2::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2599).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec2**
- namespace **activemq::wireformat::openwire::marshalsec2::v2**

## 7.421 src/main/activemq/wireformat/openwire/marshalsec2/v3/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshalsec2/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2603).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.422 src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotification File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2607).*

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.423 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2616).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.424 src/main/activemq/wireformat/openwire/marshal/v6/MessageDispatchNotification File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**  
(p. 2595).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.425 src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
```



## 7.426 src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h

---

### File Reference 4291

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2648).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.426 src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h

---

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2628).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.427 src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**  
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2640).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.428 src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
```

## 7.429 src/main/activemq/wireformat/openwire/marshal/v5/MessageldMarshaller.h

---

### File Reference 4293

```
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageldMarshaller**  
*Marshaling code for Open Wire Format for **MessageldMarshaller** (p. 2632).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.429 src/main/activemq/wireformat/openwire/marshal/v5/MessageldMarshaller.h

---

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageldMarshaller**  
*Marshaling code for Open Wire Format for **MessageldMarshaller** (p. 2636).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.430 src/main/activemq/wireformat/openwire/marshal/v6/MessageldMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageldMarshaller**

*Marshaling code for Open Wire Format for **MessageldMarshaller** (p. 2644).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.431 src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2670).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.432 src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2661).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.433 src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2657).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.434 src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2666).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.435 src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2653).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.436 src/main/activemq/wireformat/openwire/marshal/v6/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::MessageMarshaller**  
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2674).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**



7.437

src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h

File Reference

7.437 src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h

4299

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2716).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.438 src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2700).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.439 src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2708).*

## Namespaces

- namespace **activemq**

#### 7.440

**src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h**

##### File Reference

4301

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

#### 7.440 **src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h**

##### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

##### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2712).*

##### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

#### 7.441 **src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h**

##### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2704).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.442 src/main/activemq/wireformat/openwire/marshal/v6/MessagePullMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2720).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.443 src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2769).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.444 src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2749).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.445 src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.446 src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h File Reference

4305

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2761).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.446 src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2765).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.447 src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2757).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**



## 7.448 src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h File Reference

### 7.448 src/main/activemq/wireformat/openwire/marshal/v6/NetworkBridgeFilterMarshaller.h File Reference 4307

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2753).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.449 src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2891).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.450 src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2874).*

## Namespaces

- namespace **activemq**

7.451

**src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h**

**File Reference**

**4309**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.451 **src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h** **File Reference**

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2883).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.452 **src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h** **File Reference**

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2887).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.453 src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## 7.454

src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h

File Reference

4311

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2878).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.454 src/main/activemq/wireformat/openwire/marshal/v6/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2870).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.455 src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3008).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.456 src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

7.457

src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h

File Reference

4313

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2988).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.457 src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2996).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.458 src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2992).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**



7.459

src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h

File Reference

7.459 — 4315 src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3000).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.460 src/main/activemq/wireformat/openwire/marshal/v6/ProducerAckMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 3004).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.461 src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3039).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

7.462

src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h File Reference 4317

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.462 src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3019).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.463 src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3027).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.464 src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3023).*

7.465

**src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h File Reference** **4319**  
**Namespaces**

---

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.465 src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3031).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.466 src/main/activemq/wireformat/openwire/marshal/v6/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller**  
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 3035).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.467 src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.468

src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h

File Reference

4321

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3056).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.468 src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3052).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.469 src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3064).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.470 src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```



## 7.471

**src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h**

### File Reference

4323

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3047).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.471 src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3060).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.472 src/main/activemq/wireformat/openwire/marshal/v6/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 3068).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

7.473

src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h File

Reference

4325

7.473 src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3153).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.474 src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3141).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.475 src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3149).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.476

src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h File

Reference

4327

7.476 src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3161).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.477 src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3157).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.478 src/main/activemq/wireformat/openwire/marshal/v6/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller**  
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 3145).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.479 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h File Reference

### 7.479 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h File Reference

4329

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3169).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.480 src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3178).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.481 src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3174).*

## Namespaces

- namespace **activemq**



## 7.482 src/main/activemq/wireformat/openwire/marsh shal/v4/RemoveSubscriptionInfoMarshaller.h File Reference

4331

*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.482 src/main/activemq/wireformat/openwire/marsh shal/v4/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3190).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.483 src/main/activemq/wireformat/openwire/marsh shal/v5/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3186).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.484 src/main/activemq/wireformat/openwire/marshal/v6/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 3182).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.485 src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3201).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.486 src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3205).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.487 src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

## 7.488 src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h File Reference

4335

```
#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3209).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.488 src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3197).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.489 src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3217).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.490 src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h File Reference

### 7.490 src/main/activemq/wireformat/openwire/marshal/v6/ReplayCommandMarshaller.h File Reference

4337

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshalsec6::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 3213).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshalsec6**
- namespace **activemq::wireformat::openwire::marshalsec6::v6**

## 7.491 src/main/activemq/wireformat/openwire/marshalsec6/v1/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshalsec6/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3255).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.492 src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3241).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**



## 7.493 src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h

---

### File Reference 4339

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.493 src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h

---

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3250).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.494 src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h

---

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3236).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.495 src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3246).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.496 src/main/activemq/wireformat/openwire/marshal/v6/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ResponseMarshaller**  
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 3260).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.497 src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3344).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.498 src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3324).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.499 src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3340).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.500 src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3328).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.501 src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3336).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.502 src/main/activemq/wireformat/openwire/marshal/v6/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller**  
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 3332).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.503 src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3360).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.504 src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```



## 7.505

### src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h File Reference

---

4347

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3368).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.505 src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3364).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.506 src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3372).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.507 src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## 7.508

### src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h File Reference

---

4349

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3356).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.508 src/main/activemq/wireformat/openwire/marshal/v6/SessionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

#### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller**  
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 3352).*

#### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.509 src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3424).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.510 src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

7.511

src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h

File Reference

4351

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3420).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.511 src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3432).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.512 src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3436).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.513 src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

7.514

src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h

File Reference

4353

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3428).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.514 src/main/activemq/wireformat/openwire/marshal/v6/ShutdownInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 3416).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.515 src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3624).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**



7.516

src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

File Reference

4355

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.516 src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3640).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.517 src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3620).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.518 src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3632).*

7.519

src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h

File Reference

4357

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.519 src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3628).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.520 src/main/activemq/wireformat/openwire/marshal/v6/SubscriptionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 3636).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.521 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.522

src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h

### File Reference

4359

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3766).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.522 src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3770).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.523 src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utis/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3774).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.524 src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.525

src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h

File Reference

4361

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3778).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.525 src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3763).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.526 src/main/activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3781).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**



7.527

src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h

File Reference

4363

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.527 src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3793).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.528 src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3809).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.529 src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3797).*

## 7.530

src/main/activemq/wireformat/openwire/marshall/v4/TransactionInfoMarshaller.h

File Reference

4365

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.530 src/main/activemq/wireformat/openwire/marshall/v4/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshall/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3805).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.531 src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3789).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.532 src/main/activemq/wireformat/openwire/marshal/v6/TransactionInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.533

src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h

### File Reference

4367

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3801).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.533 src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3939).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.534 src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utls/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3931).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.535 src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.536

src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h

File Reference

4369

---

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3943).*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.536 src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h

## File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3935).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.537 src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3923).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**



7.538

src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h

File Reference

4371

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.538 src/main/activemq/wireformat/openwire/marshal/v6/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3927).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.539 src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3976).*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

## 7.540 src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

## Data Structures

- class **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3968).*

## 7.541

**src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h**

### File Reference

4373

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

## 7.541 src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3980).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

## 7.542 src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3972).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

## 7.543 src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

## 7.544

src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h

### File Reference

4375

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3984).*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

## 7.544 src/main/activemq/wireformat/openwire/marshal/v6/XATransactionIdMarshaller.h

### File Reference

```
#include <activemq/wireformat/openwire/marshal/v6/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

### Data Structures

- class **activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3964).*

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v6**

## 7.545 src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <memory>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

## 7.546 src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
```

## 7.547 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference 4377

---

```
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/commands/WireFormatInfo.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.547 src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatNegotiator**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.548 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

### Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

## 7.549 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

### Data Structures

- class **activemq::wireformat::openwire::utils::BooleanStream**

*Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.*



## 7.550 src/main/activemq/wireformat/openwire/Utils/HexTable.h File Reference 4379

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

## 7.550 src/main/activemq/wireformat/openwire/Utils/HexTable.h File Reference

```
#include <vector>
#include <string>
#include <activemq/Util/Config.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **activemq::wireformat::openwire::utils::HexTable**  
*The **HexTable** (p. 1947) class maps hexadecimal strings to the value of an index into the table, i.e.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

## 7.551 src/main/activemq/wireformat/openwire/Utils/MessagePropertyInterceptor.h File Reference

```
#include <activemq/Util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/Util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

## Data Structures

- class **activemq::wireformat::openwire::utils::MessagePropertyInterceptor**  
*Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

## 7.552 src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference

```
#include <cms/Destination.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <string>
#include <map>
```

## Data Structures

- class **activemq::wireformat::stomp::StompCommandConstants**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.553 src/main/activemq/wireformat/stomp/StompFrame.h File Reference

```
#include <string>
```

```
#include <string.h>
#include <map>
#include <decaf/util/Properties.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <activemq/util/Config.h>
```

## Data Structures

- class **activemq::wireformat::stomp::StompFrame**  
*A Stomp-level message frame that encloses all messages to and from the broker.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.554 src/main/activemq/wireformat/stomp/StompHelper.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::wireformat::stomp::StompHelper**  
*Utility Methods used when marshaling to and from StompFrame's.*

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.555 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/wireformat/stomp/StompHelper.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

## Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.556 src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/wireformat/stomp/StompWireFormat.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**

*Factory used to create the Stomp Wire Format instance.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

## 7.557 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/Transport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

## Data Structures

- class **activemq::wireformat::WireFormat**

*Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.*

## Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **activemq::wireformat**

## 7.558 src/main/activemq/wireformat/WireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatFactory**  
*The **WireFormatFactory** (p. 3911) is the interface that all **WireFormatFactory** (p. 3911) classes must extend.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

## 7.559 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatNegotiator**  
*Defines a **WireFormatNegotiator** (p. 3946) which allows a **WireFormat** (p. 3907) to.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

## 7.560 src/main/activemq/wireformat/WireFormatRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/wireformat/WireFormatFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **activemq::wireformat::WireFormatRegistry**  
*Registry of all **WireFormat** (p. 3907) Factories that are available to the client at run-time.*

### Namespaces

- namespace **activemq**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **activemq::wireformat**

## 7.561 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
```

### Data Structures

- class **cms::BytesMessage**

A **BytesMessage** (p. 1023) object is used to send a message containing a stream of unsigned bytes.

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.562 src/main/cms/Closeable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Closeable**

*Interface for a class that implements the close method.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.563 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::Closeable**

*Interface for a class that implements the close method.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**



## 7.564 src/main/cms/CMSException.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <cms/Config.h>
```

### Data Structures

- class **cms::CMSException**

*CMS API Exception that is the base for all exceptions thrown from CMS classes.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.565 src/main/cms/CMSProperties.h File Reference

```
#include <cms/Config.h>
#include <map>
#include <string>
#include <vector>
```

### Data Structures

- class **cms::CMSProperties**

*Interface for a Java-like properties object.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.566 src/main/cms/CMSSecurityException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::CMSSecurityException**

*This exception must be thrown when a provider rejects a user name/password submitted by a client.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.567 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stoppable.h>
#include <cms/Closeable.h>
#include <cms/Session.h>
#include <cms/ConnectionMetaData.h>
```

### Data Structures

- class **cms::Connection**

*The client's connection to its provider.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.568 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/Connection.h>
#include <cms/CMSException.h>
#include <string>
```

### Data Structures

- class **cms::ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1232) objects returned implement the CMS **Connection** (p. 1232) interface and hide the CMS Provider specific implementation details behind that interface.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.569 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::ConnectionMetaData**

*A **ConnectionMetaData** (p. 1355) object provides information describing the **Connection** (p. 1232) object.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.570 src/main/cms/DeliveryMode.h File Reference

```
#include <cms/Config.h>
```

## Data Structures

- class **cms::DeliveryMode**

*This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.571 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h>
#include <cms/Config.h>
#include <string>
```

## Data Structures

- class **cms::Destination**

*A **Destination** (p. 1688) object encapsulates a provider-specific address.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.572 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1801) that is registered with the **Connection** (p. 1232).*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.573 src/main/cms/IllegalStateException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::IllegalStateException**

*This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.574 src/main/decaf/lang/exceptions/IllegalStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.575 src/main/cms/InvalidClientIdException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidClientIdException**

*This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.576 src/main/cms/InvalidDestinationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::InvalidDestinationException**

*This exception must be thrown when a destination either is not understood by a provider or is no longer valid.*

### Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.577 src/main/cms/InvalidSelectorException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::InvalidSelectorException**

*This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.578 src/main/cms/MapMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

## Data Structures

- class **cms::MapMessage**

*A **MapMessage** (p. 2431) object is used to send a set of name-value pairs.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.579 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
```

## Data Structures

- class **cms::MessageConsumer**

*A client uses a **MessageConsumer** (p. 2550) to received messages from a destination.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.580 src/main/cms/MessageEnumeration.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::MessageEnumeration**

*Defines an object that enumerates a collection of Messages.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.581 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 3595) or **BytesMessage** (p. 1023) is being read.*



## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.582 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::MessageFormatException**

*This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.583 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

## Data Structures

- class **cms::MessageListener**

*A **MessageListener** (p. 2652) object is used to receive asynchronously delivered messages.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.584 src/main/cms/MessageNotReadableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageNotReadableException**  
*This exception must be thrown when a CMS client attempts to read a write-only message.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.585 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::MessageNotWriteableException**  
*This exception must be thrown when a CMS client attempts to write to a read-only message.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.586 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/Closeable.h>
```

```
#include <cms/CMSException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/MessageFormatException.h>
#include <cms/UnsupportedOperationException.h>
#include <cms/DeliveryMode.h>
```

## Data Structures

- class **cms::MessageProducer**

*A client uses a **MessageProducer** (p. 2681) object to send messages to a **Destination** (p. 1688).*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.587 src/main/cms/ObjectMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
```

## Data Structures

- class **cms::ObjectMessage**

*Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.588 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Queue**

*An interface encapsulating a provider-specific queue name.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.589 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::util::Queue< E >**

*A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.590 src/main/cms/QueueBrowser.h File Reference

```
#include <string>
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Queue.h>
```

```
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageEnumeration.h>
```

## Data Structures

- class **cms::QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p. 3093) without removing them.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.591 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Message.h>
#include <cms/TextMessage.h>
#include <cms/BytesMessage.h>
#include <cms/MapMessage.h>
#include <cms/StreamMessage.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <cms/Topic.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/TemporaryTopic.h>
#include <cms/TemporaryQueue.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Session**

A **Session** (p. 3305) object is a single-threaded context for producing and consuming messages.

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.592 src/main/cms/Startable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Startable**

*Interface for a class that implements the start method.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.593 src/main/cms/Stopable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Stopable**

*Interface for a class that implements the stop method.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.594 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

### Data Structures

- class **cms::StreamMessage**  
*Interface for a **StreamMessage** (p. 3595).*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.595 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::TemporaryQueue**  
*Defines a Temporary **Queue** (p. 3093) based **Destination** (p. 1688).*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.596 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

### Data Structures

- class **cms::TemporaryTopic**  
*Defines a Temporary **Topic** (p. 3757) based **Destination** (p. 1688).*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.597 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotWriteableException.h>
```

### Data Structures

- class **cms::TextMessage**  
*Interface for a text message.*

### Namespaces

- namespace **cms**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.598 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
```



```
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::Topic**

*An interface encapsulating a provider-specific topic name.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.599 src/main/cms/UnsupportedOperationException.h File Reference

```
#include <cms/Config.h>
```

```
#include <cms/CMSException.h>
```

## Data Structures

- class **cms::UnsupportedOperationException**

*This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.*

## Namespaces

- namespace **cms**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

## 7.600 src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.601 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h>
#include <apr_pools.h>
```

## Data Structures

- class **decaf::internal::AprPool**

*Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**

## 7.602 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runtime.h>
#include <apr_pools.h>
```

## Data Structures

- class **decaf::internal::DecafRuntime**

*Handles APR initialization and termination.*

## 7.603 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference 4405

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**

## 7.603 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**  
*Wrapper Around the Standard error Output facility on the current platform.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.604 src/main/decaf/internal/io/StandardInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardInputStream**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.605 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

### Data Structures

- class **decaf::internal::io::StandardOutputStream**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::io**

## 7.606 src/main/decaf/internal/net/DefaultServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::DefaultServerSocketFactory**  
*Default implementation of the Decaf ServerSocketFactory, creates ServerSocket objects with supplied options.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.607 src/main/decaf/internal/net/DefaultSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::DefaultSocketFactory**  
*SocketFactory implementation that is used to create Sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.608 src/main/decaf/internal/net/Network.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/internal/util/Resource.h>
#include <decaf/internal/util/GenericResource.h>
```

### Data Structures

- class **decaf::internal::net::Network**  
*Internal class used to manage Networking related resources and hide platform dependent calls from the higher level API.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.609 src/main/decaf/internal/net/SocketFileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FileDescriptor.h>
```

### Data Structures

- class **decaf::internal::net::SocketFileDescriptor**  
*File Descriptor type used internally by Decaf Socket objects.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.610 src/main/decaf/internal/net/ssl/DefaultSSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContext.h>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLContext**  
*Default SSLContext manager for the Decaf library.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.611 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLServerSocketFactory**

*Default implementation of the SSLServerSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.612 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
#include <string>
#include <vector>
```

### Data Structures

- class **decaf::internal::net::ssl::DefaultSSLSocketFactory**

*Default implementation of the SSLSocketFactory, this factory throws an Exception from all its create methods to indicate that SSL is not supported, this factory is used when OpenSSL is not enabled in the builds.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**

## 7.613 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLContextSpi**  
*Provides an SSLContext that wraps the OpenSSL API.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.614 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

### Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLParameters**  
*Container class for parameters that are Common to OpenSSL socket classes.*



## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.615 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocket.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocket**  
*SSLServerSocket based on OpenSSL library code.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.616 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLServerSocketFactory.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory**  
*SSLServerSocketFactory that creates Server Sockets that use OpenSSL.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.617 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocket.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocket**

*Wraps a a Normal Socket object and extends or overrides functions in that class to make use of the OpenSSL Socket API.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.618 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketException**  
*Subclass of the standard SocketException that knows how to produce an error message from the OpenSSL error stack.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.619 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLSocketFactory.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketFactory**  
*Client Socket Factory that creates SSL based client sockets using the OpenSSL library.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.620 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream**

*An output stream for reading data from an OpenSSL Socket instance.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.621 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

## Data Structures

- class **decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream**  
*OutputStream implementation used to write data to an **OpenSSLSocket** (p. 2808) instance.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::ssl**
- namespace **decaf::internal::net::ssl::openssl**

## 7.622 src/main/decaf/internal/net/tcp/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketImpl.h>
```

## 7.623 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference 415

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
#include <decaf/io/IOException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::internal::net::tcp::TcpSocket**  
*Platform-independent implementation of the socket interface.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.623 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::internal::net::tcp::TcpSocketInputStream**  
*Input stream for performing reads on a socket.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.624 src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
```

## Data Structures

- class **decaf::internal::net::tcp::TcpSocketOutputStream**  
*Output stream for performing write operations on a socket.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**
- namespace **decaf::internal::net::tcp**

## 7.625 src/main/decaf/internal/net/URLEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <string>
```

## Data Structures

- class **decaf::internal::net::URLEncoderDecoder**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.626 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/internal/net/URIType.h>
```

## Data Structures

- class **decaf::internal::net::URIHelper**  
*Helper class used by the URI classes in encoding and decoding of URI's.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.627 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::internal::net::URIType**  
*Basic type object that holds data that composes a given URI.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::net**

## 7.628 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::internal::nio::BufferFactory**  
*Factory class used by static methods in the **decaf::nio** (p. 136) package to create the various default version of the NIO interfaces.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.629 src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```



```
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::ByteArrayBuffer**

*This class defines six categories of operations upon byte buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.630 src/main/decaf/internal/nio/CharArrayBuffer.h File Reference

```
#include <decaf/nio/CharBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.631 src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference

```
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.632 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.633 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::nio::IntArrayBuffer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

### 7.634 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

#### Data Structures

- class **decaf::internal::nio::LongArrayBuffer**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

### 7.635 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/util/ByteArrayAdapter.h>
#include <decaf/lang/Pointer.h>
```

## 7.636 src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference

### Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::nio**

## 7.636 src/main/decaf/internal/security/unix/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

### Data Structures

- class **decaf::internal::security::SecureRandomImpl**  
*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::security**

## 7.637 src/main/decaf/internal/security/windows/SecureRandomImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandomSpi.h>
```

## Data Structures

- class **decaf::internal::security::SecureRandomImpl**

*Secure Random Number Generator for Unix based platforms that attempts to obtain secure bytes with high entropy from known sources.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::security**

## 7.638 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

## Data Structures

- class **decaf::internal::util::ByteArrayAdapter**

*This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.*

- union **decaf::internal::util::ByteArrayAdapter::Array**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.639 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference 4425

### 7.639 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference

```
#include <decaf/util/Config.h>
```

#### Data Structures

- class **decaf::internal::util::concurrent::ConditionImpl**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

### 7.640 src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference

```
#include <decaf/util/Config.h>
```

#### Data Structures

- class **decaf::internal::util::concurrent::MutexImpl**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.641 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::SynchronizableImpl**  
*A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.642 src/main/decaf/internal/util/concurrent/Transferer.h File Reference

```
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/util/concurrent/TimeoutException.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::Transferer< E >**  
*Shared internal API for dual stacks and queues.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**



## 7.643 src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/locks/LockSupport.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Thread.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::TransferQueue**< **E** >  
*This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.644 src/main/decaf/internal/util/concurrent/TransferStack.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

### Data Structures

- class **decaf::internal::util::concurrent::TransferStack**< **E** >

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

## 7.645 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ConditionHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.646 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::ConditionHandle**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.647 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference

### 7.647 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

#### Data Structures

- class **decaf::util::concurrent::MutexHandle**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.648 src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

#### Data Structures

- class **decaf::util::concurrent::MutexHandle**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.649 src/main/decaf/internal/util/GenericResource.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/internal/util/Resource.h>
```

## Data Structures

- class **decaf::internal::util::GenericResource**< T >  
*A Generic **Resource** (p. 3223) wraps some type and will delete it when the **Resource** (p. 3223) itself is deleted.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.650 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

## Data Structures

- class **decaf::internal::util::HexStringParser**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.651 src/main/decaf/internal/util/Resource.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::internal::util::Resource**  
*Interface for all Managed Resources in Decaf, these objects are added to the Runtime System and are destroyed at shutdown.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.652 src/main/decaf/internal/util/TimerTaskHeap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/TimerTask.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::internal::util::TimerTaskHeap**  
*A Binary Heap implemented specifically for the Timer class in Decaf Util.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::internal**
- namespace **decaf::internal::util**

## 7.653 src/main/decaf/internal/util/zip/crc32.h File Reference

### Variables

- local const unsigned long FAR **crc\_table** [TBLS][256]

### 7.653.1 Variable Documentation

7.653.1.1 local const unsigned long FAR **crc\_table**[TBLS][256]

## 7.654 src/main/decaf/internal/util/zip/deflate.h File Reference

```
#include "zutil.h"
```

## Data Structures

- struct **ct\_data\_s**
- struct **tree\_desc\_s**
- struct **internal\_state**

## Defines

- #define **GZIP**
- #define **LENGTH\_CODES** 29
- #define **LITERALS** 256
- #define **L\_CODES** (LITERALS+1+LENGTH\_CODES)
- #define **D\_CODES** 30
- #define **BL\_CODES** 19
- #define **HEAP\_SIZE** (2\*L\_CODES+1)
- #define **MAX\_BITS** 15
- #define **INIT\_STATE** 42
- #define **EXTRA\_STATE** 69
- #define **NAME\_STATE** 73
- #define **COMMENT\_STATE** 91
- #define **HCRC\_STATE** 103
- #define **BUSY\_STATE** 113
- #define **FINISH\_STATE** 666
- #define **Freq** fc.freq
- #define **Code** fc.code
- #define **Dad** dl.dad
- #define **Len** dl.len
- #define **max\_insert\_length** max\_lazy\_match
- #define **put\_byte**(s, c) {s->pending\_buf[s->pending++] = (c);}
- #define **MIN\_LOOKAHEAD** (MAX\_MATCH+MIN\_MATCH+1)
- #define **MAX\_DIST**(s) ((s)->w\_size-MIN\_LOOKAHEAD)
- #define **WIN\_INIT** MAX\_MATCH
- #define **d\_code**(dist) ((dist) < 256 ? **\_dist\_code**[dist] : **\_dist\_code**[256+((dist)>>7)])
- #define **\_tr\_tally\_lit**(s, c, flush)
- #define **\_tr\_tally\_dist**(s, distance, length, flush)

## Typedefs

- typedef struct **ct\_data\_s** **ct\_data**
- typedef struct static\_tree\_desc\_s **static\_tree\_desc**
- typedef struct **tree\_desc\_s** **tree\_desc**
- typedef **ush** **Pos**
- typedef **Pos** FAR **Posf**
- typedef unsigned **IPos**
- typedef struct **internal\_state** **deflate\_state**

## Functions

- void ZLIB\_INTERNAL \_tr\_init **OF** ((deflate\_state \*s))
- int ZLIB\_INTERNAL \_tr\_tally **OF** ((deflate\_state \*s, unsigned dist, unsigned lc))
- void ZLIB\_INTERNAL \_tr\_flush\_block **OF** ((deflate\_state \*s, charf \*buf, ulg stored\_len, int last))

## Variables

- uch ZLIB\_INTERNAL \_length\_code []
- uch ZLIB\_INTERNAL \_dist\_code []

### 7.654.1 Define Documentation

#### 7.654.1.1 #define \_tr\_tally\_dist( s, distance, length, flush )

##### Value:

```
{ uch len = (length); \
  ush dist = (distance); \
  s->d_buf[s->last_lit] = dist; \
  s->l_buf[s->last_lit++] = len; \
  dist--; \
  s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
  s->dyn_dtree[d_code(dist)].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

#### 7.654.1.2 #define \_tr\_tally\_lit( s, c, flush )

##### Value:

```
{ uch cc = (c); \
  s->d_buf[s->last_lit] = 0; \
  s->l_buf[s->last_lit++] = cc; \
  s->dyn_ltree[cc].Freq++; \
  flush = (s->last_lit == s->lit_bufsize-1); \
}
```

#### 7.654.1.3 #define BL\_CODES 19

#### 7.654.1.4 #define BUSY\_STATE 113

#### 7.654.1.5 #define Code fc.code

#### 7.654.1.6 #define COMMENT\_STATE 91

```
7.654.1.7  #define d_code( dist ) ((dist) < 256 ? _dist_code[dist] :
           _dist_code[256+((dist)>>7)])

7.654.1.8  #define D_CODES 30

7.654.1.9  #define Dad dl.dad

7.654.1.10 #define EXTRA_STATE 69

7.654.1.11 #define FINISH_STATE 666

7.654.1.12 #define Freq fc.freq

7.654.1.13 #define GZIP

7.654.1.14 #define HCRC_STATE 103

7.654.1.15 #define HEAP_SIZE (2*L_CODES+1)

7.654.1.16 #define INIT_STATE 42

7.654.1.17 #define L_CODES (LITERALS+1+LENGTH_CODES)

7.654.1.18 #define Len dl.len

7.654.1.19 #define LENGTH_CODES 29

7.654.1.20 #define LITERALS 256

7.654.1.21 #define MAX_BITS 15

7.654.1.22 #define MAX_DIST( s ) ((s)->w_size-MIN_LOOKAHEAD)

7.654.1.23 #define max_insert_length max_lazy_match

7.654.1.24 #define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)

7.654.1.25 #define NAME_STATE 73

7.654.1.26 #define put_byte( s, c ) {s->pending_buf[s->pending++] = (c);}

7.654.1.27 #define WIN_INIT MAX_MATCH

7.654.2    Typedef Documentation

7.654.2.1  typedef struct ct_data_s ct_data
```



7.654.2.2 typedef struct internal\_state deflate\_state

7.654.2.3 typedef unsigned IPos

7.654.2.4 typedef ush Pos

7.654.2.5 typedef Pos FAR Posf

7.654.2.6 typedef struct static\_tree\_desc\_s static\_tree\_desc

7.654.2.7 typedef struct tree\_desc\_s tree\_desc

### 7.654.3 Function Documentation

7.654.3.1 void ZLIB\_INTERNAL \_tr\_init OF ( (deflate\_state \*s) )

7.654.3.2 void ZLIB\_INTERNAL \_tr\_flush\_block OF ( (deflate\_state \*s, charf \*buf, ulg stored\_len, int last) )

7.654.3.3 int ZLIB\_INTERNAL \_tr\_tally OF ( (deflate\_state \*s, unsigned dist, unsigned lc) )

### 7.654.4 Variable Documentation

7.654.4.1 uch ZLIB\_INTERNAL \_dist\_code[]

7.654.4.2 uch ZLIB\_INTERNAL \_length\_code[]

## 7.655 src/main/decaf/internal/util/zip/gzguts.h File Reference

```
#include <stdio.h>
```

```
#include "zlib.h"
```

```
#include <fcntl.h>
```

### Data Structures

- struct **gz\_state**

### Defines

- #define **ZLIB\_INTERNAL**
- #define **local** static
- #define **zstrerror()** "stdio error (consult errno)"
- #define **GZBUFSIZE** 8192
- #define **GZ\_NONE** 0
- #define **GZ\_READ** 7247

- `#define GZ_WRITE 31153`
- `#define GZ_APPEND 1`
- `#define LOOK 0`
- `#define COPY 1`
- `#define GZIP 2`
- `#define GT_OFF(x) (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())`

## Typedefs

- `typedef gz_state FAR * gz_statep`

## Functions

- `voidp malloc OF ((uInt size))`
- `void free OF ((voidpf ptr))`
- `ZEXTERN gzFile ZEXPORT gzopen64 OF ((const char *, const char *))`
- `ZEXTERN z_off64_t ZEXPORT gzseek64 OF ((gzFile, z_off64_t, int))`
- `ZEXTERN z_off64_t ZEXPORT gztell64 OF ((gzFile))`
- `void ZLIB_INTERNAL gz_error OF ((gz_statep, int, const char *))`
- `unsigned ZLIB_INTERNAL gz_intmax OF ((void))`

## 7.655.1 Define Documentation

7.655.1.1 `#define COPY 1`

7.655.1.2 `#define GT_OFF( x ) (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())`

7.655.1.3 `#define GZ_APPEND 1`

7.655.1.4 `#define GZ_NONE 0`

7.655.1.5 `#define GZ_READ 7247`

7.655.1.6 `#define GZ_WRITE 31153`

7.655.1.7 `#define GZBUFSIZE 8192`

7.655.1.8 `#define GZIP 2`

7.655.1.9 `#define local static`

7.655.1.10 `#define LOOK 0`

7.655.1.11 `#define ZLIB_INTERNAL`

7.655.1.12 `#define zsterror( )` "stdio error (consult errno)"

## 7.655.2 Typedef Documentation

7.655.2.1 `typedef gz_state FAR* gz_statep`

## 7.655.3 Function Documentation

7.655.3.1 `voidp malloc OF ( (uint size) )`

7.655.3.2 `unsigned ZLIB_INTERNAL gz_intmax OF ( (void) )`

7.655.3.3 `void ZLIB_INTERNAL gz_error OF ( (gz_statep, int, const char *) )`

7.655.3.4 `ZEXTERN z_off64_t ZEXPORT gztell64 OF ( (gzFile) )`

7.655.3.5 `ZEXTERN z_off64_t ZEXPORT gzseek64 OF ( (gzFile, z_off64_t, int) )`

7.655.3.6 `ZEXTERN gzFile ZEXPORT gzopen64 OF ( (const char *, const char *) )`

7.655.3.7 `void free OF ( (voidpf ptr) )`

## 7.656 src/main/decaf/internal/util/zip/inffast.h File Reference

### Functions

- `void ZLIB_INTERNAL inflate_fast OF ((z_streamp strm, unsigned start))`

## 7.656.1 Function Documentation

7.656.1.1 `void ZLIB_INTERNAL inflate_fast OF ( (z_streamp strm, unsigned start) )`

## 7.657 src/main/decaf/internal/util/zip/inffixed.h File Reference

## 7.658 src/main/decaf/internal/util/zip/inflate.h File Reference

### Data Structures

- `struct inflate_state`

### Defines

- `#define GUNZIP`

## Enumerations

- enum **inflate\_mode** {  
    **HEAD**, **FLAGS**, **TIME**, **OS**,  
    **EXLEN**, **EXTRA**, **NAME**, **COMMENT**,  
    **HCRC**, **DICTID**, **DICT**, **TYPE**,  
    **TYPEDO**, **STORED**, **COPY\_**, **COPY**,  
    **TABLE**, **LENLENS**, **CODELENS**, **LEN\_**,  
    **LEN**, **LENEXT**, **DIST**, **DISTEXT**,  
    **MATCH**, **LIT**, **CHECK**, **LENGTH**,  
    **DONE**, **BAD**, **MEM**, **SYNC** }

### 7.658.1 Define Documentation

#### 7.658.1.1 #define GUNZIP

### 7.658.2 Enumeration Type Documentation

#### 7.658.2.1 enum inflate\_mode

##### Enumerator:

**HEAD**  
**FLAGS**  
**TIME**  
**OS**  
**EXLEN**  
**EXTRA**  
**NAME**  
**COMMENT**  
**HCRC**  
**DICTID**  
**DICT**  
**TYPE**  
**TYPEDO**  
**STORED**  
**COPY\_**  
**COPY**  
**TABLE**  
**LENLENS**  
**CODELENS**

*LEN\_*  
*LEN*  
*LENEXT*  
*DIST*  
*DISTEXT*  
*MATCH*  
*LIT*  
*CHECK*  
*LENGTH*  
*DONE*  
*BAD*  
*MEM*  
*SYNC*

## 7.659 src/main/decaf/internal/util/zip/inftrees.h File Reference

### Data Structures

- struct **code**

### Defines

- #define **ENOUGH\_LENS** 852
- #define **ENOUGH\_DISTS** 592
- #define **ENOUGH** (ENOUGH\_LENS+ENOUGH\_DISTS)

### Enumerations

- enum **codetype** { **CODES**, **LENS**, **DISTS** }

### Functions

- int ZLIB\_INTERNAL inflate\_table **OF** ((**codetype** type, unsigned short FAR \*lens, unsigned codes, **code** FAR \*FAR \*table, unsigned FAR \*bits, unsigned short FAR \*work))

### 7.659.1 Define Documentation

7.659.1.1 `#define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)`

7.659.1.2 `#define ENOUGH_DISTS 592`

7.659.1.3 `#define ENOUGH_LENS 852`

### 7.659.2 Enumeration Type Documentation

7.659.2.1 `enum codetype`

Enumerator:

***CODES***

***LENS***

***DISTS***

### 7.659.3 Function Documentation

7.659.3.1 `int ZLIB_INTERNAL inflate_table OF ( (codetype type, unsigned short FAR *lens, unsigned codes, code FAR *FAR *table, unsigned FAR *bits, unsigned short FAR *work) )`

## 7.660 src/main/decaf/internal/util/zip/trees.h File Reference

### Variables

- local const **ct\_data static\_ltree** [L\_CODES+2]
- local const **ct\_data static\_dtree** [D\_CODES]
- const **uch** ZLIB\_INTERNAL **\_dist\_code** [DIST\_CODE\_LEN]
- const **uch** ZLIB\_INTERNAL **\_length\_code** [MAX\_MATCH-MIN\_MATCH+1]
- local const int **base\_length** [LENGTH\_CODES]
- local const int **base\_dist** [D\_CODES]

### 7.660.1 Variable Documentation

7.660.1.1 `const uch ZLIB_INTERNAL _dist_code[DIST_CODE_LEN]`

Initial value:

```
{
  0,  1,  2,  3,  4,  4,  5,  5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  8,
  8,  8,  8,  8,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,
```

```
7.660.1.2  const uch ZLIB_INTERNAL_length_code[MAX_MATCH-MIN_MATCH+1]
```

**Initial value:**

```
{
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 10, 10, 11, 11, 12, 12, 12, 12,
13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16,
17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28
}
```

### 7.660.1.3 local const int base\_dist[D\_CODES]

**Initial value:**

[illegible]

#### 7.660.1.4 local const int base length[LENGTH\_CODES]

**Initial value:**

```
{
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
}
```

#### 7.660.1.5 local const ct\_data static\_dtree[D\_CODES]

**Initial value:**

```
{
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
}
```

#### 7.660.1.6 local const ct\_data static\_ltree[L\_CODES+2]

### 7.661 src/main/decaf/internal/util/zip/zconf.h File Reference

```
#include <decaf/util/Config.h>
```

#### Defines

- #define **const**
- #define **MAX\_MEM\_LEVEL** 9
- #define **MAX\_WBITS** 15
- #define **OF**(args) ()
- #define **ZEXTERN** extern
- #define **ZEXPORT**
- #define **ZEXPORTVA**
- #define **FAR**
- #define **SEEK\_SET** 0
- #define **SEEK\_CUR** 1
- #define **SEEK\_END** 2
- #define **z\_off\_t** long
- #define **z\_off64\_t** z\_off\_t

#### Typedefs

- typedef unsigned char **Byte**
- typedef unsigned int **ulnt**
- typedef unsigned long **uLong**
- typedef **Byte** FAR **Bytef**



- typedef char FAR **charf**
- typedef int FAR **intf**
- typedef **uInt** FAR **uIntf**
- typedef **uLong** FAR **uLongf**
- typedef **Byte** const \* **voidpc**
- typedef **Byte** FAR \* **voidpf**
- typedef **Byte** \* **voidp**

### 7.661.1 Define Documentation

7.661.1.1 #define **const**

7.661.1.2 #define **FAR**

7.661.1.3 #define **MAX\_MEM\_LEVEL** 9

7.661.1.4 #define **MAX\_WBITS** 15

7.661.1.5 #define **OF( args )** ()

7.661.1.6 #define **SEEK\_CUR** 1

7.661.1.7 #define **SEEK\_END** 2

7.661.1.8 #define **SEEK\_SET** 0

7.661.1.9 #define **z\_off64\_t z\_off\_t**

7.661.1.10 #define **z\_off\_t** long

7.661.1.11 #define **ZEXPORT**

7.661.1.12 #define **ZEXPORTVA**

7.661.1.13 #define **ZEXTERN** extern

### 7.661.2 Typedef Documentation

7.661.2.1 typedef unsigned char **Byte**

7.661.2.2 typedef **Byte** FAR **Bytef**

7.661.2.3 typedef char FAR **charf**

7.661.2.4 typedef int FAR **intf**

7.661.2.5 `typedef unsigned int uInt`

7.661.2.6 `typedef uInt FAR uIntf`

7.661.2.7 `typedef unsigned long uLong`

7.661.2.8 `typedef uLong FAR uLongf`

7.661.2.9 `typedef Byte* voidp`

7.661.2.10 `typedef Byte const* voidpc`

7.661.2.11 `typedef Byte FAR* voidpf`

## 7.662 `src/main/decaf/internal/util/zip/zlib.h` File Reference

```
#include "zconf.h"
```

### Data Structures

- struct `z_stream_s`
- struct `gz_header_s`
- struct `internal_state`

### Defines

- `#define ZLIB_VERSION "1.2.5"`
- `#define ZLIB_VERNUM 0x1250`
- `#define ZLIB_VER_MAJOR 1`
- `#define ZLIB_VER_MINOR 2`
- `#define ZLIB_VER_REVISION 5`
- `#define ZLIB_VER_SUBREVISION 0`
- `#define Z_NO_FLUSH 0`
- `#define Z_PARTIAL_FLUSH 1`
- `#define Z_SYNC_FLUSH 2`
- `#define Z_FULL_FLUSH 3`
- `#define Z_FINISH 4`
- `#define Z_BLOCK 5`
- `#define Z_TREES 6`
- `#define Z_OK 0`
- `#define Z_STREAM_END 1`
- `#define Z_NEED_DICT 2`
- `#define Z_ERRNO (-1)`
- `#define Z_STREAM_ERROR (-2)`
- `#define Z_DATA_ERROR (-3)`

- `#define Z_MEM_ERROR (-4)`
- `#define Z_BUF_ERROR (-5)`
- `#define Z_VERSION_ERROR (-6)`
- `#define Z_NO_COMPRESSION 0`
- `#define Z_BEST_SPEED 1`
- `#define Z_BEST_COMPRESSION 9`
- `#define Z_DEFAULT_COMPRESSION (-1)`
- `#define Z_FILTERED 1`
- `#define Z_HUFFMAN_ONLY 2`
- `#define Z_RLE 3`
- `#define Z_FIXED 4`
- `#define Z_DEFAULT_STRATEGY 0`
- `#define Z_BINARY 0`
- `#define Z_TEXT 1`
- `#define Z_ASCII Z_TEXT`
- `#define Z_UNKNOWN 2`
- `#define Z_DEFLATED 8`
- `#define Z_NULL 0`
- `#define zlib_version zlibVersion()`
- `#define deflateInit(strm, level) deflateInit_((strm), (level), ZLIB_VERSION, sizeof(z_stream))`
- `#define inflateInit(strm) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))`
- `#define deflateInit2(strm, level, method, windowBits, memLevel, strategy)`
- `#define inflateInit2(strm, windowBits) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))`
- `#define inflateBackInit(strm, windowBits, window)`

## Typedefs

- `typedef voidpf alloc_func OF ((voidpf opaque, ulint items, ulint size))`
- `typedef struct z_stream_s z_stream`
- `typedef z_stream FAR * z_streamp`
- `typedef struct gz_header_s gz_header`
- `typedef gz_header FAR * gz_headerp`
- `typedef voidp gzFile`

## Functions

- `ZEXTERN const char *ZEXPORT zlibVersion OF ((void))`
- `ZEXTERN int ZEXPORT deflate OF ((z_streamp strm, int flush))`
- `ZEXTERN int ZEXPORT deflateEnd OF ((z_streamp strm))`
- `ZEXTERN int ZEXPORT deflateSetDictionary OF ((z_streamp strm, const Bytef *dictionary, ulintdictLength))`
- `ZEXTERN int ZEXPORT deflateCopy OF ((z_streamp dest, z_streamp source))`
- `ZEXTERN int ZEXPORT deflateParams OF ((z_streamp strm, int level, int strategy))`

- ZEXTERN int ZEXPORT deflateTune **OF** ((**z\_stream** strm, int good\_length, int max\_lazy, int nice\_length, int max\_chain))
- ZEXTERN **uLong** ZEXPORT deflateBound **OF** ((**z\_stream** strm, **uLong** sourceLen))
- ZEXTERN int ZEXPORT deflatePrime **OF** ((**z\_stream** strm, int bits, int value))
- ZEXTERN int ZEXPORT deflateSetHeader **OF** ((**z\_stream** strm, **gz\_headerp** head))
- ZEXTERN int ZEXPORT inflateReset2 **OF** ((**z\_stream** strm, int windowBits))
- ZEXTERN int ZEXPORT inflateBack **OF** ((**z\_stream** strm, in\_func in, void FAR \*in\_desc, out\_func out, void FAR \*out\_desc))
- ZEXTERN int ZEXPORT compress **OF** ((**Bytef** \*dest, **uLongf** \*destLen, const **Bytef** \*source, **uLong** sourceLen))
- ZEXTERN int ZEXPORT compress2 **OF** ((**Bytef** \*dest, **uLongf** \*destLen, const **Bytef** \*source, **uLong** sourceLen, int level))
- ZEXTERN **uLong** ZEXPORT compressBound **OF** ((**uLong** sourceLen))
- ZEXTERN **gzFile** ZEXPORT gzdopen **OF** ((int fd, const char \*mode))
- ZEXTERN int ZEXPORT gzbuffer **OF** ((**gzFile** file, unsigned size))
- ZEXTERN int ZEXPORT gzsetparams **OF** ((**gzFile** file, int level, int strategy))
- ZEXTERN int ZEXPORT gzread **OF** ((**gzFile** file, **voidp** buf, unsigned len))
- ZEXTERN int ZEXPORT gzwrite **OF** ((**gzFile** file, **voidpc** buf, unsigned len))
- ZEXTERN int ZEXPORTVA gzprintf **OF** ((**gzFile** file, const char \*format,...))
- ZEXTERN int ZEXPORT gzputs **OF** ((**gzFile** file, const char \*s))
- ZEXTERN char \*ZEXPORT gzgets **OF** ((**gzFile** file, char \*buf, int len))
- ZEXTERN int ZEXPORT gzputc **OF** ((**gzFile** file, int c))
- ZEXTERN int ZEXPORT gzgetc **OF** ((**gzFile** file))
- ZEXTERN int ZEXPORT gzungetc **OF** ((int c, **gzFile** file))
- ZEXTERN int ZEXPORT gzflush **OF** ((**gzFile** file, int flush))
- ZEXTERN const char \*ZEXPORT gzerror **OF** ((**gzFile** file, int \*errnum))
- ZEXTERN **uLong** ZEXPORT Adler32 **OF** ((**uLong** adler, const **Bytef** \*buf, **uInt** len))
- ZEXTERN **uLong** ZEXPORT crc32 **OF** ((**uLong** crc, const **Bytef** \*buf, **uInt** len))
- ZEXTERN int ZEXPORT deflateInit\_ **OF** ((**z\_stream** strm, int level, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateInit\_ **OF** ((**z\_stream** strm, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT deflateInit2\_ **OF** ((**z\_stream** strm, intlevel, intmethod, int windowBits, int memLevel, int strategy, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateInit2\_ **OF** ((**z\_stream** strm, intwindowBits, const char \*version, int stream\_size))
- ZEXTERN int ZEXPORT inflateBackInit\_ **OF** ((**z\_stream** strm, int windowBits, unsigned char FAR \*window, const char \*version, int stream\_size))
- ZEXTERN **gzFile** ZEXPORT gzopen **OF** ((const char \*, const char \*))
- ZEXTERN **z\_off\_t** ZEXPORT gzseek **OF** ((**gzFile**, **z\_off\_t**, int))
- ZEXTERN **z\_off\_t** ZEXPORT gtell **OF** ((**gzFile**))
- ZEXTERN **uLong** ZEXPORT Adler32\_combine **OF** ((**uLong**, **uLong**, **z\_off\_t**))
- ZEXTERN const char \*ZEXPORT zError **OF** ((int))
- ZEXTERN int ZEXPORT inflateSyncPoint **OF** ((**z\_stream**))
- ZEXTERN int ZEXPORT inflateUndermine **OF** ((**z\_stream**, int))

### 7.662.1 Define Documentation

7.662.1.1 `#define deflateInit( strm, level ) deflateInit_((strm), (level), ZLIB_VERSION, sizeof(z_stream))`

7.662.1.2 `#define deflateInit2( strm, level, method, windowBits, memLevel, strategy )`

**Value:**

```
deflateInit2_((strm), (level), (method), (windowBits), (memLevel), \
               (strategy),                ZLIB_VERSION, sizeof(z_stream))
```

7.662.1.3 `#define inflateBackInit( strm, windowBits, window )`

**Value:**

```
inflateBackInit_((strm), (windowBits), (window), \
                  ZLIB_VERSION, sizeof(z_stream))
```

7.662.1.4 `#define inflateInit( strm ) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))`

7.662.1.5 `#define inflateInit2( strm, windowBits ) inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))`

7.662.1.6 `#define Z_ASCII Z_TEXT`

7.662.1.7 `#define Z_BEST_COMPRESSION 9`

7.662.1.8 `#define Z_BEST_SPEED 1`

7.662.1.9 `#define Z_BINARY 0`

7.662.1.10 `#define Z_BLOCK 5`

7.662.1.11 `#define Z_BUF_ERROR (-5)`

7.662.1.12 `#define Z_DATA_ERROR (-3)`

7.662.1.13 `#define Z_DEFAULT_COMPRESSION (-1)`

7.662.1.14 `#define Z_DEFAULT_STRATEGY 0`

7.662.1.15 `#define Z_DEFLATED 8`

7.662.1.16 `#define Z_ERRNO (-1)`

7.662.1.17 `#define Z_FILTERED 1`

7.662.1.18 `#define Z_FINISH 4`

7.662.1.19 `#define Z_FIXED 4`

7.662.1.20 `#define Z_FULL_FLUSH 3`

7.662.1.21 `#define Z_HUFFMAN_ONLY 2`

7.662.1.22 `#define Z_MEM_ERROR (-4)`

7.662.1.23 `#define Z_NEED_DICT 2`

7.662.1.24 `#define Z_NO_COMPRESSION 0`

7.662.1.25 `#define Z_NO_FLUSH 0`

7.662.1.26 `#define Z_NULL 0`

7.662.1.27 `#define Z_OK 0`

7.662.1.28 `#define Z_PARTIAL_FLUSH 1`

7.662.1.29 `#define Z_RLE 3`

7.662.1.30 `#define Z_STREAM_END 1`

7.662.1.31 `#define Z_STREAM_ERROR (-2)`

7.662.1.32 `#define Z_SYNC_FLUSH 2`

7.662.1.33 `#define Z_TEXT 1`

7.662.1.34 `#define Z_TREES 6`

7.662.1.35 `#define Z_UNKNOWN 2`

7.662.1.36 `#define Z_VERSION_ERROR (-6)`

7.662.1.37 `#define ZLIB_VER_MAJOR 1`

7.662.1.38 `#define ZLIB_VER_MINOR 2`

7.662.1.39 `#define ZLIB_VER_REVISION 5`

7.662.1.40 `#define ZLIB_VER_SUBREVISION 0`

7.662.1.41 `#define ZLIB_VERNUM 0x1250`

7.662.1.42 `#define zlib_version zlibVersion()`

7.662.1.43 `#define ZLIB_VERSION "1.2.5"`

## 7.662.2 Typedef Documentation

7.662.2.1 `typedef struct gz_header_s gz_header`

7.662.2.2 `typedef gz_header FAR* gz_headerp`

7.662.2.3 `typedef voidp gzFile`

7.662.2.4 `ZEXTERN uLong ZEXPORT crc32_combine64 OF ( (voidpf opaque, ulInt items, ulInt size) )`

7.662.2.5 `typedef struct z_stream_s z_stream`

7.662.2.6 `typedef z_stream FAR* z_streamp`

## 7.662.3 Function Documentation

7.662.3.1 `ZEXTERN const char* ZEXPORT zlibVersion OF ( (void) )`

7.662.3.2 `ZEXTERN int ZEXPORT inflateUndermine OF ( (z_streamp, int) )`

7.662.3.3 `ZEXTERN int ZEXPORT inflateSyncPoint OF ( (z_streamp) )`

7.662.3.4 `ZEXTERN const char* ZEXPORT zError OF ( (int) )`

7.662.3.5 `ZEXTERN uLong ZEXPORT Adler32_combine OF ( (uLong, uLong, z_off_t) )`

7.662.3.6 `ZEXTERN z_off_t ZEXPORT gzTell OF ( (gzFile) )`

7.662.3.7 `ZEXTERN z_off_t ZEXPORT gzSeek OF ( (gzFile, z_off_t, int) )`

7.662.3.8 `ZEXTERN gzFile ZEXPORT gzOpen OF ( (const char *, const char *) )`

7.662.3.9 `ZEXTERN int ZEXPORT inflateBackInit_ OF ( (z_streamp strm, int windowBits, unsigned char FAR *window, const char *version, int stream_size) )`

7.662.3.10 `ZEXTERN int ZEXPORT inflateInit2_ OF ( (z_streamp strm, int windowBits, const char *version, int stream_size) )`

7.662.3.11 `ZEXTERN int ZEXPORT deflateInit2_ OF ( (z_streamp strm, int level, int method, int windowBits, int memLevel, int strategy, const char *version, int stream_size) )`

- 7.662.3.12 ZEXTERN int ZEXPORT inflateInit OF ( (z\_stream strm, const char \*version, int stream\_size) )
- 7.662.3.13 ZEXTERN int ZEXPORT deflateInit OF ( (z\_stream strm, int level, const char \*version, int stream\_size) )
- 7.662.3.14 ZEXTERN uLong ZEXPORT crc32 OF ( (uLong crc, const Bytef \*buf, ulint len) )
- 7.662.3.15 ZEXTERN uLong ZEXPORT Adler32 OF ( (uLong Adler, const Bytef \*buf, ulint len) )
- 7.662.3.16 ZEXTERN const char\* ZEXPORT gzerror OF ( (gzFile file, int \*errnum) )
- 7.662.3.17 ZEXTERN int ZEXPORT gzflush OF ( (gzFile file, int flush) )
- 7.662.3.18 ZEXTERN int ZEXPORT gzungetc OF ( (int c, gzFile file) )
- 7.662.3.19 ZEXTERN int ZEXPORT gzgetc OF ( (gzFile file) )
- 7.662.3.20 ZEXTERN int ZEXPORT gzputc OF ( (gzFile file, int c) )
- 7.662.3.21 ZEXTERN char\* ZEXPORT gzgets OF ( (gzFile file, char \*buf, int len) )
- 7.662.3.22 ZEXTERN int ZEXPORT gzputs OF ( (gzFile file, const char \*s) )
- 7.662.3.23 ZEXTERN int ZEXPORTVA gzprintf OF ( (gzFile file, const char \*format,...) )
- 7.662.3.24 ZEXTERN int ZEXPORT gzwrite OF ( (gzFile file, voidpc buf, unsigned len) )
- 7.662.3.25 ZEXTERN int ZEXPORT gzread OF ( (gzFile file, voidp buf, unsigned len) )
- 7.662.3.26 ZEXTERN int ZEXPORT gzsetparams OF ( (gzFile file, int level, int strategy) )
- 7.662.3.27 ZEXTERN int ZEXPORT gzbuffer OF ( (gzFile file, unsigned size) )
- 7.662.3.28 ZEXTERN gzFile ZEXPORT gzdopen OF ( (int fd, const char \*mode) )
- 7.662.3.29 ZEXTERN uLong ZEXPORT compressBound OF ( (uLong sourceLen) )
- 7.662.3.30 ZEXTERN int ZEXPORT compress2 OF ( (Bytef \*dest, uLongf \*destLen, const Bytef \*source, uLong sourceLen, int level) )
- 7.662.3.31 ZEXTERN int ZEXPORT compress OF ( (Bytef \*dest, uLongf \*destLen, const Bytef \*source, uLong sourceLen) )
- 7.662.3.32 ZEXTERN int ZEXPORT inflateBack OF ( (z\_stream strm, in\_func in, void FAR \*in\_desc, out\_func out, void FAR \*out\_desc) )



- 7.662.3.33 ZEXTERN int ZEXPORT inflateReset2 OF ( (z\_streamp strm, int windowBits) )
- 7.662.3.34 ZEXTERN int ZEXPORT deflateSetHeader OF ( (z\_streamp strm, gz\_headerp head) )
- 7.662.3.35 ZEXTERN int ZEXPORT deflatePrime OF ( (z\_streamp strm, int bits, int value) )
- 7.662.3.36 ZEXTERN uLong ZEXPORT deflateBound OF ( (z\_streamp strm, uLong sourceLen) )
- 7.662.3.37 ZEXTERN int ZEXPORT deflateTune OF ( (z\_streamp strm, int good\_length, int max\_lazy, int nice\_length, int max\_chain) )
- 7.662.3.38 ZEXTERN int ZEXPORT deflateParams OF ( (z\_streamp strm, int level, int strategy) )
- 7.662.3.39 ZEXTERN int ZEXPORT deflateCopy OF ( (z\_streamp dest, z\_streamp source) )
- 7.662.3.40 ZEXTERN int ZEXPORT deflateSetDictionary OF ( (z\_streamp strm, const Bytef \*dictionary, ulndictLength) )
- 7.662.3.41 ZEXTERN int ZEXPORT deflateEnd OF ( (z\_streamp strm) )
- 7.662.3.42 ZEXTERN int ZEXPORT deflate OF ( (z\_streamp strm, int flush) )

## 7.663 src/main/decaf/internal/util/zip/zutil.h File Reference

```
#include "zlib.h"
```

### Defines

- #define **ZLIB\_INTERNAL**
- #define **local** static
- #define **ERR\_MSG**(err) **z\_errmsg**[Z\_NEED\_DICT-(err)]
- #define **ERR\_RETURN**(strm, err) return (strm->msg = (char\*)ERR\_MSG(err), (err))
- #define **DEF\_WBITS** MAX\_WBITS
- #define **DEF\_MEM\_LEVEL** 8
- #define **STORED\_BLOCK** 0
- #define **STATIC\_TREES** 1
- #define **DYN\_TREES** 2
- #define **MIN\_MATCH** 3
- #define **MAX\_MATCH** 258
- #define **PRESET\_DICT** 0x20
- #define **OS\_CODE** 0x03
- #define **F\_OPEN**(name, mode) fopen((name), (mode))

- `#define Assert(cond, msg)`
- `#define Trace(x)`
- `#define Tracev(x)`
- `#define Tracevv(x)`
- `#define Tracec(c, x)`
- `#define Tracecv(c, x)`
- `#define ZALLOC(strm, items, size) (*((strm)->zalloc))((strm)->opaque, (items), (size))`
- `#define ZFREE(strm, addr) (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))`
- `#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}`

## Typedefs

- `typedef unsigned char uch`
- `typedef uch FAR uchf`
- `typedef unsigned short ush`
- `typedef ush FAR ushf`
- `typedef unsigned long ulg`

## Functions

- `ZEXTERN uLong ZEXPORT adler32_combine64 OF ((uLong, uLong, z_off_t))`
- `void ZLIB_INTERNAL zmemcpy OF ((Bytef *dest, const Bytef *source, uInt len))`
- `int ZLIB_INTERNAL zmemcmp OF ((const Bytef *s1, const Bytef *s2, uInt len))`
- `void ZLIB_INTERNAL zmemzero OF ((Bytef *dest, uInt len))`
- `voidpf ZLIB_INTERNAL zcalloc OF ((voidpf opaque, unsigned items, unsigned size))`
- `void ZLIB_INTERNAL zcfree OF ((voidpf opaque, voidpf ptr))`

## Variables

- `const char *const z_errmsg [10]`

## 7.663.1 Define Documentation

7.663.1.1 `#define Assert( cond, msg )`

7.663.1.2 `#define DEF_MEM_LEVEL 8`

7.663.1.3 `#define DEF_WBITS MAX_WBITS`

7.663.1.4 `#define DYN_TREES 2`

- 7.663.1.5 `#define ERR_MSG( err ) z_errmsg[Z_NEED_DICT-(err)]`
- 7.663.1.6 `#define ERR_RETURN( strm, err ) return (strm->msg = (char*)ERR_MSG(err), (err))`
- 7.663.1.7 `#define F_OPEN( name, mode ) fopen((name), (mode))`
- 7.663.1.8 `#define local static`
- 7.663.1.9 `#define MAX_MATCH 258`
- 7.663.1.10 `#define MIN_MATCH 3`
- 7.663.1.11 `#define OS_CODE 0x03`
- 7.663.1.12 `#define PRESET_DICT 0x20`
- 7.663.1.13 `#define STATIC_TREES 1`
- 7.663.1.14 `#define STORED_BLOCK 0`
- 7.663.1.15 `#define Trace( x )`
- 7.663.1.16 `#define Tracec( c, x )`
- 7.663.1.17 `#define Tracecv( c, x )`
- 7.663.1.18 `#define Tracev( x )`
- 7.663.1.19 `#define Tracevv( x )`
- 7.663.1.20 `#define TRY_FREE( s, p ) { if (p) ZFREE(s, p); }`
- 7.663.1.21 `#define ZALLOC( strm, items, size ) (*((strm)->zalloc)((strm)->opaque, (items), (size))`
- 7.663.1.22 `#define ZFREE( strm, addr ) (*((strm)->zfree)((strm)->opaque, (voidpf)(addr))`
- 7.663.1.23 `#define ZLIB_INTERNAL`
- 7.663.2 **Typedef Documentation**
  - 7.663.2.1 `typedef unsigned char uch`
  - 7.663.2.2 `typedef uch FAR uchf`
  - 7.663.2.3 `typedef unsigned long ulg`

7.663.2.4 typedef unsigned short ush

7.663.2.5 typedef ush FAR ushf

### 7.663.3 Function Documentation

7.663.3.1 ZEXTERN uLong ZEXPORT Adler32\_combine64 OF ( (uLong, uLong, z\_off\_t) )

7.663.3.2 void ZLIB\_INTERNAL zcfree OF ( (voidpf opaque, voidpf ptr) )

7.663.3.3 voidpf ZLIB\_INTERNAL zcalloc OF ( (voidpf opaque, unsigned items, unsigned size) )

7.663.3.4 void ZLIB\_INTERNAL zmemzero OF ( (Bytef \*dest, uInt len) )

7.663.3.5 int ZLIB\_INTERNAL zmemcmp OF ( (const Bytef \*s1, const Bytef \*s2, uInt len) )

7.663.3.6 void ZLIB\_INTERNAL zmemcpy OF ( (Bytef \*dest, const Bytef \*source, uInt len) )

### 7.663.4 Variable Documentation

7.663.4.1 const char\* const z\_errmsg[10]

## 7.664 src/main/decaf/io/BlockingByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
```

```
#include <vector>
```

### Data Structures

- class **decaf::io::BlockingByteArrayInputStream**

*This is a blocking version of a byte buffer stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.665 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::io::BufferedInputStream**

*A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.666 src/main/decaf/io/BufferedOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::io::BufferedOutputStream**

*Wrapper around another output stream that buffers output before writing to the target output stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.667 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

### Data Structures

- class **decaf::io::ByteArrayInputStream**

*A **ByteArrayInputStream** (p. 984) contains an internal buffer that contains bytes that may be read from the stream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.668 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <utility>
```

### Data Structures

- class **decaf::io::ByteArrayOutputStream**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.669 src/main/decaf/io/DataInput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::DataInput**

*The **DataInput** (p. 1523) interface provides for reading bytes from a binary stream and reconstructing from them data in any of the C++ primitive types.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.670 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::DataInputStream**

*A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.671 src/main/decaf/io/DataOutput.h File Reference

```
#include <decaf/util/Config.h>
#include <vector>
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::io::DataOutput**

*The **DataOutput** (p. 1541) interface provides for converting data from any of the C++ primitive types to a series of bytes and writing these bytes to a binary stream.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.672 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <string>
```



## Data Structures

- class **decaf::io::DataOutputStream**

*A data output stream lets an application write primitive Java data types to an output stream in a portable way.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.673 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::EOFException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.674 src/main/decaf/io/FileDescriptor.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::io::FileDescriptor**

*This class serves as an opaque wrapper around an underlying OS level resource that can be used as a source / sink for byte level data such as sockets and files.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.675 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::io::FilterInputStream**

*A **FilterInputStream** (p. 1854) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.676 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::io::FilterOutputStream**

*This class is the superclass of all classes that filter output streams.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.677 src/main/decaf/io/Flushable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::Flushable**  
*A **Flushable** (p. 1899) is a destination of data that can be flushed.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.678 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::io::InputStream**  
*A base class that must be implemented by all classes wishing to provide a class that reads in a stream of bytes.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.679 src/main/decaf/io/InputStreamReader.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/io/Reader.h>
```

## Data Structures

- class **decaf::io::InputStreamReader**

*An **InputStreamReader** (p. 2013) is a bridge from byte streams to character streams.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.680 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::InterruptedIOException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.681 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::io::IOException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.682 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

### Data Structures

- class **decaf::io::OutputStream**  
*Base interface for any class that wants to represent an output stream of bytes.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.683 src/main/decaf/io/OutputStreamWriter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/Writer.h>
```

### Data Structures

- class **decaf::io::OutputStreamWriter**  
*A class for turning a character stream into a byte stream.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.684 src/main/decaf/io/PushbackInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/FilterInputStream.h>
```

### Data Structures

- class **decaf::io::PushbackInputStream**  
*A **PushbackInputStream** (p. 3086) adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.685 src/main/decaf/io/Reader.h File Reference

```
#include <string>
#include <decaf/lang/Readable.h>
```

## 7.686 src/main/decaf/io/UnsupportedEncodingException.h File Reference 4465

```
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

### Data Structures

- class **decaf::io::Reader**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.686 src/main/decaf/io/UnsupportedEncodingException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::io::UnsupportedEncodingException**  
*Thrown when the the Character Encoding is not supported.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**

## 7.687 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::io::UTFDataFormatException**

*Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.688 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <vector>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/io/Flushable.h>
#include <decaf/lang/Appendable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

## Data Structures

- class **decaf::io::Writer**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**

## 7.689 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```



## Data Structures

- class **decaf::lang::Appendable**

*An object to which char sequences and values can be appended.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.690 src/main/decaf/lang/ArrayPointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/System.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

## Data Structures

- class **decaf::lang::ArrayPointer< T, REFCOUNTER >**

*Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.*

- struct **decaf::lang::ArrayPointer< T, REFCOUNTER >::ArrayData**

- class **decaf::lang::ArrayPointerComparator< T, R >**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the value of the actual pointer to the array being contained in this **ArrayPointer** (p. 697).*

- struct **std::less< decaf::lang::ArrayPointer< T > >**

*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **std**

## Functions

- template<typename T , typename R , typename U >  
bool **decaf::lang::operator==** (const ArrayPointer< T, R > &left, const U \*right)
- template<typename T , typename R , typename U >  
bool **decaf::lang::operator==** (const U \*left, const ArrayPointer< T, R > &right)
- template<typename T , typename R , typename U >  
bool **decaf::lang::operator!=** (const ArrayPointer< T, R > &left, const U \*right)
- template<typename T , typename R , typename U >  
bool **decaf::lang::operator!=** (const U \*left, const ArrayPointer< T, R > &right)

## 7.691 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>
#include <decaf/lang/Comparable.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Boolean**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.692 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

## Data Structures

- class **decaf::lang::Byte**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.693 src/main/decaf/lang/Character.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

## Data Structures

- class **decaf::lang::Character**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.694 src/main/decaf/lang/CharSequence.h File Reference

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::CharSequence**

*A **CharSequence** (p. 1107) is a readable sequence of char values.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.695 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Comparable< T >**

*This interface imposes a total ordering on the objects of each class that implements it.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.696 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/Number.h>
```

```
#include <decaf/lang/exceptions/NumberFormatException.h>
```

```
#include <string>
```

## Data Structures

- class **decaf::lang::Double**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.697 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/lang/exceptions/ExceptionDefines.h>
#include <decaf/util/Config.h>
#include <stdarg.h>
#include <sstream>
```

## Data Structures

- class **decaf::lang::Exception**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.698 src/main/decaf/lang/exceptions/ClassCastException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::ClassCastException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.699 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalArgumentException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.700 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalMonitorStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.701 src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IllegalThreadStateException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.702 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.703 src/main/decaf/lang/exceptions/InterruptedException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::InterruptedException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.704 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.705 src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::lang::exceptions::NoSuchElementException**



## **7.706 src/main/decaf/lang/exceptions/NullPointerException.h File Reference 4475**

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## **7.706 src/main/decaf/lang/exceptions/NullPointerException.h File Reference**

```
#include <decaf/lang/Exception.h>
```

### **Data Structures**

- class **decaf::lang::exceptions::NullPointerException**

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## **7.707 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference**

```
#include <decaf/lang/Exception.h>
```

### **Data Structures**

- class **decaf::lang::exceptions::NumberFormatException**

### **Namespaces**

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.708 src/main/decaf/lang/exceptions/RuntimeException.h File Reference

```
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::lang::exceptions::RuntimeException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

## 7.709 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

### Data Structures

- class **decaf::lang::Float**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.710 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
#include <decaf/lang/exceptions/NumberFormatException.h>
```

## Data Structures

- class **decaf::lang::Integer**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.711 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
```

## Data Structures

- class **decaf::lang::Iterable< E >**  
*Implementing this interface allows an object to be cast to an **Iterable** (p. 2112) type for generic collections API calls.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.712 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

## Data Structures

- class **decaf::lang::Long**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.713 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Math**

*The class **Math** (p. 2455) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.714 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Number**

*The abstract class **Number** (p. 2786) is the superclass of classes **Byte** (p. 918), **Double** (p. 1751), **Float** (p. 1865), **Integer** (p. 2038), **Long** (p. 2377), and **Short** (p. 3380).*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.715 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/ClassCastException.h>
#include <decaf/util/concurrent/atomic/AtomicRefCounter.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

## Data Structures

- struct **decaf::lang::STATIC\_CAST\_TOKEN**
- struct **decaf::lang::DYNAMIC\_CAST\_TOKEN**
- class **decaf::lang::Pointer< T, REFCOUNTER >**  
*Decaf's implementation of a Smart **Pointer** (p. 2896) that is a template on a Type and is **Thread** (p. 3707) Safe if the default Reference Counter is used.*
- class **decaf::lang::PointerComparator< T, R >**  
*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2896) instance.*
- struct **std::less< decaf::lang::Pointer< T > >**  
*An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **std**

## Functions

- `template<typename T , typename R , typename U >`  
`bool decaf::lang::operator== (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool decaf::lang::operator== (const U *left, const Pointer< T, R > &right)`
- `template<typename T , typename R , typename U >`  
`bool decaf::lang::operator!= (const Pointer< T, R > &left, const U *right)`
- `template<typename T , typename R , typename U >`  
`bool decaf::lang::operator!= (const U *left, const Pointer< T, R > &right)`

## 7.716 src/main/decaf/lang/Readable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::lang::Readable**  
*A **Readable** (p. 3106) is a source of characters.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**
- namespace **decaf::lang**

## 7.717 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Runnable**  
*Interface for a runnable object - defines a task that can be run by a thread.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.718 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Runtime**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.719 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Number.h>  
#include <decaf/lang/Comparable.h>  
#include <decaf/lang/exceptions/NumberFormatException.h>  
#include <string>
```

## Data Structures

- class **decaf::lang::Short**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**

## 7.720 src/main/decaf/lang/String.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

### Data Structures

- class **decaf::lang::String**

*The **String** (p. 3610) class represents an immutable sequence of chars.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.721 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/internal/AprPool.h>
#include <string>
```

### Data Structures

- class **decaf::lang::System**

*The **System** (p. 3670) class provides static methods for accessing system level resources and performing some system dependent tasks such as looking up environment values and copying memory and arrays.*

### Namespaces

- namespace **decaf**



*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.722 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/exceptions/IllegalThreadStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::Thread**  
*A **Thread** (p. 3707) is a concurrent unit of execution.*
- class **decaf::lang::Thread::UncaughtExceptionHandler**  
*Interface for handlers invoked when a **Thread** (p. 3707) abruptly terminates due to an uncaught exception.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

## 7.723 src/main/decaf/lang/ThreadGroup.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::lang::ThreadGroup**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.724 src/main/decaf/lang/Throwable.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::lang::Throwable**

*This class represents an error that has occurred.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**

## 7.725 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

## Data Structures

- class **decaf::net::BindException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.726 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

### Data Structures

- class **decaf::net::ConnectException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.727 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::HttpRetryException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.728 src/main/decaf/net/Inet4Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

## Data Structures

- class **decaf::net::Inet4Address**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.729 src/main/decaf/net/Inet6Address.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

## Data Structures

- class **decaf::net::Inet6Address**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.730 src/main/decaf/net/InetAddress.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/ArrayPointer.h>
```

## Data Structures

- class **decaf::net::InetAddress**  
*Represents an IP address.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.731 src/main/decaf/net/InetSocketAddress.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketAddress.h>
#include <string>
```

## Data Structures

- class **decaf::net::InetSocketAddress**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.732 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::net::MalformedURLException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

### 7.733 src/main/decaf/net/NoRouteToHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

#### Data Structures

- class **decaf::net::NoRouteToHostException**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

### 7.734 src/main/decaf/net/PortUnreachableException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

#### Data Structures

- class **decaf::net::PortUnreachableException**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

### 7.735 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

#### Data Structures

- class **decaf::net::ProtocolException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.736 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImpl.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
#include <string>
```

## Data Structures

- class **decaf::net::ServerSocket**

*This class implements server sockets.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.737 src/main/decaf/net/ServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/InetAddress.h>
```

## Data Structures

- class **decaf::net::ServerSocketFactory**

*Class used to create Server Sockets, subclasses can be created that create certain types of server sockets according to specific policies.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.738 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/InetAddress.h>
#include <decaf/net/SocketImplFactory.h>
#include <decaf/net/SocketException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/UnknownHostException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::net::Socket**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**



## 7.739 src/main/decaf/net/SocketAddress.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketAddress**  
*Base class for protocol specific **Socket** (p. 3445) addresses.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.740 src/main/decaf/net/SocketError.h File Reference

```
#include <string>  
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::net::SocketError**  
*Static utility class to simplify handling of error codes for socket operations.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.741 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

### Data Structures

- class **decaf::net::SocketException**  
*Exception for errors when manipulating sockets.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.742 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/net/UnknownHostException.h>
```

## Data Structures

- class **decaf::net::SocketFactory**

*The **SocketFactory** (p. 3467) is used to create **Socket** (p. 3445) objects and can be sub-classed to provide other types of Sockets or Sockets with varying configurations.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.743 src/main/decaf/net/SocketImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/FileDescriptor.h>
#include <decaf/net/SocketException.h>
#include <decaf/net/SocketTimeoutException.h>
#include <decaf/net/SocketOptions.h>
#include <string>
```

## Data Structures

- class **decaf::net::SocketImpl**  
*Acts as a base class for all physical **Socket** (p. 3445) implementations.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.744 src/main/decaf/net/SocketImplFactory.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::net::SocketImplFactory**  
*Factory class interface for a Factory that creates SocketImpl objects.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.745 src/main/decaf/net/SocketOptions.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::net::SocketOptions**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.746 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InterruptedIOException.h>
```

### Data Structures

- class **decaf::net::SocketTimeoutException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.747 src/main/decaf/net/ssl/SSLContext.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ssl/SSLContextSpi.h>
```

### Data Structures

- class **decaf::net::ssl::SSLContext**  
*Represents on implementation of the Secure **Socket** (p. 3445) Layer for streaming based sockets.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.748 src/main/decaf/net/ssl/SSLContextSpi.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/SecureRandom.h>
```

## Data Structures

- class **decaf::net::ssl::SSLContextSpi**

*Defines the interface that should be provided by an **SSLContext** (p. 3489) provider.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.749 src/main/decaf/net/ssl/SSLParameters.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
#include <vector>
```

## Data Structures

- class **decaf::net::ssl::SSLParameters**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.750 src/main/decaf/net/ssl/SSLServerSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocket.h>
```

## Data Structures

- class **decaf::net::ssl::SSLServerSocket**

*Represents a server socket that is used to accept connections from clients using the Secure Sockets protocol or the Top Level Security protocol.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.751 src/main/decaf/net/ssl/SSLServerSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/ServerSocketFactory.h>
#include <vector>
#include <string>
```

## Data Structures

- class **decaf::net::ssl::SSLServerSocketFactory**  
*Factory class interface that provides methods to create SSL Server Sockets.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.752 src/main/decaf/net/ssl/SSLSocket.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/Socket.h>
#include <decaf/net/ssl/SSLParameters.h>
```

## Data Structures

- class **decaf::net::ssl::SSLSocket**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.753 src/main/decaf/net/ssl/SSLSocketFactory.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketFactory.h>
#include <vector>
#include <string>
```

## Data Structures

- class **decaf::net::ssl::SSLSocketFactory**  
*Factory class interface for a **SocketFactory** (p. 3467) that can create **SSLSocket** (p. 3506) objects.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**
- namespace **decaf::net::ssl**

## 7.754 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::net::UnknownHostException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.755 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::net::UnknownServiceException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::net**

## 7.756 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/net/MalformedURLException.h>
#include <decaf/net/URL.h>
#include <decaf/internal/net/URIType.h>
#include <string>
```

## Data Structures

- class **decaf::net::URI**

*This class represents an instance of a **URI** (p. 3853) as defined by RFC 2396.*



## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.757 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::net::URISyntaxException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.758 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

## Data Structures

- class **decaf::net::URL**  
*Class **URL** (p. 3891) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.759 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

### Data Structures

- class **decaf::net::URLDecoder**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.760 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

### Data Structures

- class **decaf::net::URLEncoder**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::net**

## 7.761 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/nio/InvalidMarkException.h>
```

### Data Structures

- class **decaf::nio::Buffer**  
*A container for data of a specific primitive type.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.762 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::nio::BufferOverflowException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.763 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::nio::BufferUnderflowException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.764 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

### Data Structures

- class **decaf::nio::ByteBuffer**

*This class defines six categories of operations upon byte buffers:*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.765 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Appendable.h>
```

### Data Structures

- class **decaf::nio::CharBuffer**

*This class defines four categories of operations upon character buffers:*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.766 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::DoubleBuffer**  
*This class defines four categories of operations upon double buffers:*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.767 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::FloatBuffer**

*This class defines four categories of operations upon float buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.768 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::IntBuffer**

*This class defines four categories of operations upon int buffers:*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::nio**

## 7.769 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

## Data Structures

- class **decaf::nio::InvalidMarkException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.770 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::LongBuffer**  
*This class defines four categories of operations upon long long buffers:*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.771 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

## Data Structures

- class **decaf::nio::ReadOnlyBufferException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.772 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

## Data Structures

- class **decaf::nio::ShortBuffer**  
*This class defines four categories of operations upon short buffers:*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::nio**

## 7.773 src/main/decaf/security/auth/x500/X500Principal.h File Reference

```
#include <string>
#include <vector>
#include <decaf/security/Principal.h>
```



```
#include <decaf/util/Map.h>
#include <decaf/io/InputStream.h>
```

## Data Structures

- class **decaf::security::auth::x500::X500Principal**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::auth**
- namespace **decaf::security::auth::x500**

## 7.774 src/main/decaf/security/cert/Certificate.h File Reference

```
#include <vector>
#include <decaf/util/Config.h>
#include <decaf/security/InvalidKeyException.h>
#include <decaf/security/NoSuchAlgorithmException.h>
#include <decaf/security/SignatureException.h>
#include <decaf/security/cert/CertificateEncodingException.h>
#include <decaf/security/cert/CertificateException.h>
```

## Data Structures

- class **decaf::security::cert::Certificate**  
*Base interface for all identity certificates.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

### 7.775 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

#### Data Structures

- class **decaf::security::cert::CertificateEncodingException**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

### 7.776 src/main/decaf/security/cert/CertificateException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

#### Data Structures

- class **decaf::security::cert::CertificateException**

#### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

### 7.777 src/main/decaf/security/cert/CertificateExpiredException.h File Reference

```
#include <decaf/util/Config.h>
```

## 7.778 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

4509

```
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateExpiredException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.778 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/security/cert/CertificateException.h>
```

### Data Structures

- class **decaf::security::cert::CertificateNotYetValidException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.779 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/security/cert/CertificateException.h>
```

## Data Structures

- class **decaf::security::cert::CertificateParsingException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.780 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
```

## Data Structures

- class **decaf::security::cert::X509Certificate**  
*Base interface for all identity certificates.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**
- namespace **decaf::security::cert**

## 7.781 src/main/decaf/security/GeneralSecurityException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::security::GeneralSecurityException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.782 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

## Data Structures

- class **decaf::security::InvalidKeyException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.783 src/main/decaf/security/Key.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::security::Key**  
*The **Key** (p. 2253) interface is the top-level interface for all keys.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.784 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::KeyException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.785 src/main/decaf/security/KeyManagementException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/KeyException.h>
```

### Data Structures

- class **decaf::security::KeyManagementException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.786 src/main/decaf/security/NoSuchAlgorithmException.h File Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

## Data Structures

- class **decaf::security::NoSuchAlgorithmException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.787 src/main/decaf/security/NoSuchProviderException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

## Data Structures

- class **decaf::security::NoSuchProviderException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.788 src/main/decaf/security/Principal.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::security::Principal**  
*Base interface for a principal, which can represent an individual or organization.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.789 src/main/decaf/security/PublicKey.h File Reference

```
#include <decaf/security/Key.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::security::PublicKey**  
*A public key.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.790 src/main/decaf/security/SecureRandom.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Random.h>
#include <decaf/security/SecureRandomSpi.h>
#include <memory>
```

## Data Structures

- class **decaf::security::SecureRandom**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**



## 7.791 src/main/decaf/security/SecureRandomSpi.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::security::SecureRandomSpi**  
*Interface class used by Security Service Providers to implement a source of secure random bytes.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.792 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/security/GeneralSecurityException.h>
```

### Data Structures

- class **decaf::security::SignatureException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::security**

## 7.793 src/main/decaf/util/AbstractCollection.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

```
#include <decaf/lang/exceptions/NullPointerException.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

```
#include <decaf/lang/Iterable.h>
```

```
#include <decaf/util/Iterator.h>
#include <decaf/util/Collection.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractCollection**< E >

*This class provides a skeletal implementation of the **Collection** (p. 1155) interface, to minimize the effort required to implement this interface.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.794 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractList**< E >

*This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.795 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Map.h>
#include <decaf/util/Set.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractMap< K, V, COMPARATOR >**  
*This class provides a skeletal implementation of the **Map** (p. 2419) interface, to minimize the effort required to implement this interface.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.796 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

```
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Queue.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractQueue**< E >

*This class provides skeletal implementations of some **Queue** (p. 3094) operations.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.797 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractList.h>
#include <memory>
```

## Data Structures

- class **decaf::util::AbstractSequentialList**< E >

*This class provides a skeletal implementation of the **List** (p. 2296) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.798 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Set.h>
#include <memory>
```

### Data Structures

- class **decaf::util::AbstractSet**< **E** >

*This class provides a skeletal implementation of the **Set** (p. 3379) interface to minimize the effort required to implement this interface.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.799 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

## Data Structures

- class **decaf::util::Collection**< **E** >

*The root interface in the collection hierarchy.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.800 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <algorithm>
```

```
#include <functional>
```

## Data Structures

- class **decaf::util::Comparator**< **T** >

*A comparison function, which imposes a total ordering on some collection of objects.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.801 src/main/decaf/util/comparators/Less.h File Reference

```
#include <decaf/util/Comparator.h>
```

## Data Structures

- class **decaf::util::comparators::Less**< **E** >

*Simple **Less** (p. 2287) **Comparator** (p. 1189) that compares to elements to determine if the first is less than the second.*

## 7.802 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference 4521

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::comparators**

## 7.802 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**  
*A boolean value that may be updated atomically.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.803 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <string>
```

### Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**  
*An int value that may be updated atomically.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

### 7.804 src/main/decaf/util/concurrent/atomic/AtomicRefCounter.h File Reference

```
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
```

## Data Structures

- class **decaf::util::concurrent::atomic::AtomicRefCounter**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

### 7.805 src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Long.h>  
#include <apr_atomic.h>
```

## Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference< T >**  
*An Pointer reference that may be updated atomically.*



## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

## 7.806 src/main/decaf/util/concurrent/BlockingQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

## Data Structures

- class **decaf::util::concurrent::BlockingQueue< E >**  
*A **decaf::util::Queue** (p. 3094) that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.807 src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.808 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::Callable< V >**  
*A task that returns a result and may throw an exception.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.809 src/main/decaf/util/concurrent/CancellationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::CancellationException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.810 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

### Defines

- #define **WAIT\_INFINITE** 0xFFFFFFFF  
*The synchronized macro defines a mechanism for synchronizing a section of code.*
- #define **synchronized(W)**

#### 7.810.1 Define Documentation

##### 7.810.1.1 #define synchronized( W )

###### Value:

```
if(false){} \
else \
for( decaf::util::concurrent::Lock lock_W(W); \
    lock_W.isLocked(); lock_W.unlock() )
```

##### 7.810.1.2 #define WAIT\_INFINITE 0xFFFFFFFF

The synchronized macro defines a mechanism for synchronizing a section of code.

The macro must be passed an object that implements the Synchronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compilers restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Synchronizable X;

```
somefunction() { synchronized(X) (p. 4511) { // Do something that needs synchronizing.
} }
```

## 7.811 src/main/decaf/util/concurrent/ConcurrentMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

### Data Structures

- class **decaf::util::concurrent::ConcurrentMap**< K, V, COMPARATOR >  
*Interface for a **Map** (p. 2419) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 2419) interface.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.812 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/ConcurrentMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

### Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap**< K, V, COMPARATOR >  
***Map** (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.813 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::CountDownLatch**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.814 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/TimeUnit.h>
```

## Data Structures

- class **decaf::util::concurrent::Delayed**  
*A mix-in style interface for marking objects that should be acted upon after a given delay.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.815 src/main/decaf/util/concurrent/ExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::concurrent::ExecutionException**

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.816 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

## Data Structures

- class **decaf::util::concurrent::Executor**

*An object that executes submitted **decaf.lang.Runnable** (p. 3264) tasks.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.817 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Executor.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

## Data Structures

- class **decaf::util::concurrent::ExecutorService**

*An **Executor** (p. 1831) that provides methods to manage termination and methods that can produce a **Future** (p. 1929) for tracking progress of one or more asynchronous tasks.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.818 src/main/decaf/util/concurrent/Future.h File Reference

## Data Structures

- class **decaf::util::concurrent::Future< V >**

*A **Future** (p. 1929) represents the result of an asynchronous computation.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.819 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::Lock**  
*A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.820 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/concurrent/locks/Condition.h>
```

## Data Structures

- class **decaf::util::concurrent::locks::Lock**  
***Lock** (p. 2336) implementations provide more extensive locking operations than can be obtained using synchronized statements.*



## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.821 src/main/decaf/util/concurrent/locks/Condition.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
```

## Data Structures

- class **decaf::util::concurrent::locks::Condition**  
***Condition** (p. 1220) factors out the **Mutex** (p. 2736) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 2336) implementations.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.822 src/main/decaf/util/concurrent/locks/LockSupport.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::locks::LockSupport**

*Basic thread blocking primitives for creating locks and other synchronization classes.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.823 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

### Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**

*A **ReadWriteLock** (p. 3117) maintains a pair of associated locks, one for read-only operations and one for writing.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.824 src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/locks/Lock.h>
#include <decaf/lang/Pointer.h>
```

## Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**  
*A reentrant mutual exclusion **Lock** (p. 2336) with extended capabilities.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

## 7.825 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::Mutex**  
***Mutex** (p. 2736) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.826 src/main/decaf/util/concurrent/PooledThread.h File Reference

```
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/logging/LoggerDefines.h>
```

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

### Data Structures

- class **decaf::util::concurrent::PooledThread**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.827 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::concurrent::PooledThreadListener**  
*Abstract Listener Interface for users of **ThreadPool** (p. 3718).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.828 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## 7.829 [src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference](#)

### Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.829 [src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference](#)

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

### Data Structures

- class **decaf::util::concurrent::RejectedExecutionHandler**  
*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.830 [src/main/decaf/util/concurrent/Semaphore.h File Reference](#)

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <memory>
```

## Data Structures

- class **decaf::util::concurrent::Semaphore**  
*A counting semaphore.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.831 src/main/decaf/util/concurrent/Synchronizable.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::Synchronizable**  
*The interface for all synchronizable objects (that is, objects that can be locked and unlocked).*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.832 src/main/decaf/util/concurrent/SynchronousQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <vector>
```

## Data Structures

- class **decaf::util::concurrent::SynchronousQueue**< **E** >  
*A **blocking queue** (p. 804) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.*
- class **decaf::util::concurrent::SynchronousQueue**< **E** >::EmptyIterator

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.833 src/main/decaf/util/concurrent/TaskListener.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::TaskListener**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.834 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

```
#include <decaf/util/Config.h>
```

## Data Structures

- class **decaf::util::concurrent::ThreadFactory**  
*public interface **ThreadFactory** (p. 3716)*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.835 src/main/decaf/util/concurrent/ThreadPool.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/PooledThread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/concurrent/TaskListener.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <vector>
```

## Data Structures

- class **decaf::util::concurrent::ThreadPool**  
*Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.836 src/main/decaf/util/concurrent/TimeoutException.h File Reference

```
#include <decaf/lang/Exception.h>
```



## Data Structures

- class **decaf::util::concurrent::TimeoutException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.837 src/main/decaf/util/concurrent/TimeUnit.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

## Data Structures

- class **decaf::util::concurrent::TimeUnit**  
*A **TimeUnit** (p. 3748) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

## 7.838 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

### Data Structures

- class **decaf::util::Date**  
*Wrapper class around a time value in milliseconds.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.839 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

### Data Structures

- class **decaf::util::Iterator< T >**  
*Defines an object that can be used to iterate over the elements of a collection.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.840 src/main/decaf/util/List.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/util/ListIterator.h>
```

## Data Structures

- class **decaf::util::List**< **E** >  
*An ordered collection (also known as a sequence).*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.841 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::util::ListIterator**< **E** >  
*An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.842 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/StreamHandler.h>
#include <decaf/util/logging/SimpleFormatter.h>
#include <decaf/io/IOException.h>
#include <decaf/internal/io/StandardErrorOutputStream.h>
```

### Data Structures

- class **decaf::util::logging::ConsoleHandler**  
*This **Handler** (p. 1941) publishes log records to System.err.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.843 src/main/decaf/util/logging/EventManager.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::EventManager**  
***EventManager** (p. 1792) objects can be attached to Handlers to process any error that occur on a **Handler** (p. 1941) during Logging.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.844 src/main/decaf/util/logging/Filter.h File Reference

```
#include <decaf/util/logging/LogRecord.h>
```

### Data Structures

- class **decaf::util::logging::Filter**

A **Filter** (p. 1853) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.845 src/main/decaf/util/logging/Formatter.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/util/logging/Handler.h>
```

### Data Structures

- class **decaf::util::logging::Formatter**

A **Formatter** (p. 1927) provides support for formatting LogRecords.

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.846 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>  
#include <decaf/lang/Exception.h>
```

```
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/Level.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <string>
```

## Data Structures

- class **decaf::util::logging::Handler**

*A **Handler** (p. 1941) object takes log messages from a **Logger** (p. 2345) and exports them.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.847 src/main/decaf/util/logging/Level.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::util::logging::Level**

*The **Level** (p. 2290) class defines a set of standard logging levels that can be used to control logging output.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.848 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/util/logging/LogManager.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <list>
#include <string>
#include <stdarg.h>
```

### Data Structures

- class **decaf::util::logging::Logger**

A **Logger** (p. 2345) object is used to log messages for a specific system or application component.

### Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.849 src/main/decaf/util/logging/LoggerCommon.h File Reference

### Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

## Enumerations

- enum **decaf::util::logging::Levels** {  
**decaf::util::logging::Off**, **decaf::util::logging::Null**, **decaf::util::logging::Markblock**,  
**decaf::util::logging::Debug**,  
**decaf::util::logging::Info**, **decaf::util::logging::Warn**, **decaf::util::logging::Error**,  
**decaf::util::logging::Fatal**,  
**decaf::util::logging::Throwing** }

*Defines an enumeration for logging levels.*

## 7.850 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>
#include <sstream>
```

## Defines

- #define **LOGDECAF\_DECLARE**(loggerName) static **decaf::util::logging::SimpleLogger** loggerName;
- #define **LOGDECAF\_INITIALIZE**(loggerName, className, loggerFamily) **decaf::util::logging::SimpleLogger** className::loggerName(loggerFamily);
- #define **LOGDECAF\_DECLARE\_LOCAL**(loggerName) **decaf::util::logging::Logger** loggerName;
- #define **LOGDECAF\_DEBUG**(logger, message) logger.debug(\_\_FILE\_\_, \_\_LINE\_\_, message);
- #define **LOGDECAF\_DEBUG\_1**(logger, message, value)
- #define **LOGDECAF\_INFO**(logger, message) logger.info(\_\_FILE\_\_, \_\_LINE\_\_, message);
- #define **LOGDECAF\_ERROR**(logger, message) logger.error(\_\_FILE\_\_, \_\_LINE\_\_, message);
- #define **LOGDECAF\_WARN**(logger, message) logger.warn(\_\_FILE\_\_, \_\_LINE\_\_, message);
- #define **LOGDECAF\_FATAL**(logger, message) logger.fatal(\_\_FILE\_\_, \_\_LINE\_\_, message);

### 7.850.1 Define Documentation

7.850.1.1 #define **LOGDECAF\_DEBUG**( *logger, message* ) logger.debug(\_\_FILE\_\_, \_\_LINE\_\_, message);

7.850.1.2 #define **LOGDECAF\_DEBUG\_1**( *logger, message, value* )

**Value:**



```

;
{
    std::ostream ostream;
    ostream << message << value;
    logger.debug(__FILE__, __LINE__, ostream.str());
}

```

7.850.1.3 `#define LOGDECAF_DECLARE( loggerName ) static  
decaf::util::logging::SimpleLogger loggerName;`

7.850.1.4 `#define LOGDECAF_DECLARE_LOCAL( loggerName  
) decaf::util::logging::Logger loggerName;`

7.850.1.5 `#define LOGDECAF_ERROR( logger, message ) logger.error(__FILE__, __LINE__,  
message);`

7.850.1.6 `#define LOGDECAF_FATAL( logger, message ) logger.fatal(__FILE__, __LINE__,  
message);`

7.850.1.7 `#define LOGDECAF_INFO( logger, message ) logger.info(__FILE__, __LINE__,  
message);`

7.850.1.8 `#define LOGDECAF_INITIALIZE( loggerName, className, loggerFamily  
) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);`

7.850.1.9 `#define LOGDECAF_WARN( logger, message ) logger.warn(__FILE__, __LINE__,  
message);`

## 7.851 src/main/decaf/util/logging/LoggerHierarchy.h File Reference

### Data Structures

- class **decaf::util::logging::LoggerHierarchy**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor  
license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.852 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
```

```
#include <list>
#include <string>
#include <vector>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

## Data Structures

- class **decaf::util::logging::LogManager**

*There is a single global **LogManager** (p. 2363) object that is used to maintain a set of shared state about Loggers and log services.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::lang**
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.853 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Level.h>
#include <decaf/util/Config.h>
#include <memory>
#include <string>
```

## Data Structures

- class **decaf::util::logging::LogRecord**

***LogRecord** (p. 2370) objects are used to pass logging requests between the logging framework and individual log Handlers.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.854 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **decaf::util::logging::LogWriter**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.855 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

## Data Structures

- class **decaf::util::logging::MarkBlockLogger**  
*Defines a class that can be used to mark the entry and exit from scoped blocks.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.856 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

```
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::PropertiesChangeListener**  
*Defines the interface that classes can use to listen for change events on **Properties** (p. 3072).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.857 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <string>
```

### Data Structures

- class **decaf::util::logging::SimpleFormatter**  
*Print a brief summary of the **LogRecord** (p. 2370) in a human readable format.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.858 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::SimpleLogger**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.859 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/Config.h>
```

### Data Structures

- class **decaf::util::logging::StreamHandler**  
*Stream based logging **Handler** (p. 1941).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::io**
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.860 src/main/decaf/util/logging/XMLFormatter.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/LogRecord.h>
```

### Data Structures

- class **decaf::util::logging::XMLFormatter**  
*Format a **LogRecord** (p. 2370) into a standard XML format.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::logging**

## 7.861 src/main/decaf/util/Map.h File Reference

```
#include <functional>
#include <vector>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

### Data Structures

- class **decaf::util::Map< K, V, COMPARATOR >**  
***Map** (p. 2419) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*
- class **decaf::util::Map< K, V, COMPARATOR >::Entry**

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.862 src/main/decaf/util/PriorityQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Collection.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Comparator.h>
#include <decaf/util/comparators/Less.h>
#include <decaf/lang/Math.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <memory>
```

### Data Structures

- class **decaf::util::PriorityQueue**< **E** >  
*An unbounded priority queue based on a binary heap algorithm.*
- class **decaf::util::PriorityQueue**< **E** >::PriorityQueueIterator
- class **decaf::util::PriorityQueue**< **E** >::ConstPriorityQueueIterator

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.863 src/main/decaf/util/Properties.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/Pointer.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::util::Properties**

*Java-like properties class for mapping string names to string values.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::io**
- namespace **decaf::util**

## 7.864 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <vector>
#include <cmath>
```

## Data Structures

- class **decaf::util::Random**

***Random** (p. 3100) Value Generator which is used to generate a stream of pseudorandom numbers.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**



## 7.865 src/main/decaf/util/Set.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
```

### Data Structures

- class **decaf::util::Set**< E >  
*A collection that contains no duplicate elements.*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.866 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
```

### Data Structures

- class **decaf::util::StlList**< E >

*List* (p. 2296) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

- class **decaf::util::StlList**< **E** >::StlListIterator
- class **decaf::util::StlList**< **E** >::ConstStlListIterator

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.867 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

## Data Structures

- class **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** >  
*Map* (p. 2419) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.868 src/main/decaf/util/StlQueue.h File Reference

```
#include <list>
#include <vector>
#include <decaf/util/Iterator.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::StlQueue**< T >  
*The **Queue** (p. 3094) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*
- class **decaf::util::StlQueue**< T >::QueueIterator

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.869 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractSet.h>
```

## Data Structures

- class **decaf::util::StlSet**< E >  
***Set** (p. 3379) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.*
- class **decaf::util::StlSet**< E >::SetIterator
- class **decaf::util::StlSet**< E >::ConstSetIterator

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.870 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

### Data Structures

- class **decaf::util::StringTokenizer**

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.871 src/main/decaf/util/Timer.h File Reference

```
#include <memory>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

### Data Structures

- class **decaf::util::Timer**

*A facility for threads to schedule tasks for future execution in a background thread.*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**

## 7.872 src/main/decaf/util/TimerTask.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
```

## Data Structures

- class **decaf::util::TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 3730).*

## Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::util**

## 7.873 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <apr_uuid.h>
#include <string>
```

## Data Structures

- class **decaf::util::UUID**  
*A class that represents an immutable universally unique identifier (**UUID** (p. 3900)).*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**

## 7.874 src/main/decaf/util/zip/Adler32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

## Data Structures

- class **decaf::util::zip::Adler32**  
*Clas that can be used to compute an Adler-32 **Checksum** (p. 1114) for a data stream.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.875 src/main/decaf/util/zip/CheckedInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterInputStream.h>
```

## Data Structures

- class **decaf::util::zip::CheckedInputStream**  
*An implementation of a **FilterInputStream** that will maintain a **Checksum** (p. 1114) of the bytes read, the **Checksum** (p. 1114) can then be used to verify the integrity of the input stream.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.876 src/main/decaf/util/zip/CheckedOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
#include <decaf/io/FilterOutputStream.h>
```

## Data Structures

- class **decaf::util::zip::CheckedOutputStream**  
*An implementation of a `FilterOutputStream` that will maintain a **Checksum** (p. 1114) of the bytes written, the **Checksum** (p. 1114) can then be used to verify the integrity of the output stream.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.877 src/main/decaf/util/zip/Checksum.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

## Data Structures

- class **decaf::util::zip::Checksum**  
*An interface used to represent **Checksum** (p. 1114) values in the Zip package.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.878 src/main/decaf/util/zip/CRC32.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/zip/Checksum.h>
```

## Data Structures

- class **decaf::util::zip::CRC32**  
*Class that can be used to compute a CRC-32 checksum for a data stream.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.879 src/main/decaf/util/zip/DataFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

## Data Structures

- class **decaf::util::zip::DataFormatException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**



## 7.880 src/main/decaf/util/zip/Deflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::Deflater**

*This class compresses data using the DEFLATE algorithm (see specification).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.881 src/main/decaf/util/zip/DeflaterOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Deflater.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::DeflaterOutputStream**

*Provides a FilterOutputStream instance that compresses the data before writing it to the wrapped OutputStream.*

### Namespaces

- namespace **decaf**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*

- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.882 src/main/decaf/util/zip/Inflater.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/zip/DataFormatException.h>
#include <vector>
```

### Data Structures

- class **decaf::util::zip::Inflater**  
*This class uncompresses data that was compressed using the DEFLATE algorithm (see specification).*

### Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.883 src/main/decaf/util/zip/InflaterInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/zip/Inflater.h>
#include <vector>
```

## Data Structures

- class **decaf::util::zip::InflaterInputStream**  
*A FilterInputStream that decompresses data read from the wrapped InputStream instance.*

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

## 7.884 src/main/decaf/util/zip/ZipException.h File Reference

```
#include <decaf/io/IOException.h>
```

## Data Structures

- class **decaf::util::zip::ZipException**

## Namespaces

- namespace **decaf**  
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.*
- namespace **decaf::util**
- namespace **decaf::util::zip**

# Index

~AbstractCollection	241	
decaf::util::AbstractCollection, 149		activemq::wireformat::openwire::marshal::v3::ActiveMQBytesM
~AbstractList	221	
decaf::util::AbstractList, 162		activemq::wireformat::openwire::marshal::v4::ActiveMQBytesM
~AbstractMap	229	
decaf::util::AbstractMap, 163		activemq::wireformat::openwire::marshal::v5::ActiveMQBytesM
~AbstractQueue	233	
decaf::util::AbstractQueue, 164		activemq::wireformat::openwire::marshal::v6::ActiveMQBytesM
~AbstractSequentialList	237	
decaf::util::AbstractSequentialList, 168		~ActiveMQCPP
~AbstractSet		activemq::library::ActiveMQCPP, 292
decaf::util::AbstractSet, 169		~ActiveMQConnection
~AbstractTransportFactory		activemq::core::ActiveMQConnection,
activemq::transport::AbstractTransportFactory, 249		
	171	~ActiveMQConnectionFactory
~ActiveMQAckHandler		activemq::core::ActiveMQConnectionFactory,
activemq::core::ActiveMQAckHandler,	267	
	172	~ActiveMQConnectionMetaData
~ActiveMQBlobMessage		activemq::core::ActiveMQConnectionMetaData,
activemq::commands::ActiveMQBlobMessage,	276	
	174	~ActiveMQConsumer
~ActiveMQBlobMessageMarshaller		activemq::core::ActiveMQConsumer, 284
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller,	290	~ActiveMQDestination
	183	activemq::commands::ActiveMQDestination,
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller,	290	
	191	~ActiveMQDestinationMarshaller
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	309	
	178	activemq::wireformat::openwire::marshal::v1::ActiveMQDestina
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller,	321	
	187	activemq::wireformat::openwire::marshal::v2::ActiveMQDestina
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller,	305	
	195	activemq::wireformat::openwire::marshal::v3::ActiveMQDestina
activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller,	313	
	199	activemq::wireformat::openwire::marshal::v4::ActiveMQDestina
~ActiveMQBytesMessage		activemq::wireformat::openwire::marshal::v5::ActiveMQDestina
activemq::commands::ActiveMQBytesMessage, 317		
	204	activemq::wireformat::openwire::marshal::v6::ActiveMQDestina
~ActiveMQBytesMessageMarshaller	325	
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller,	329	~ActiveMQException
	225	activemq::exceptions::ActiveMQException,
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller,	329	

- ~ActiveMQMapMessage 438
  - activemq::commands::ActiveMQMapMessage, 438
  - activemq::core::ActiveMQProducer, 442
- ~ActiveMQMapMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 450
  - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 454
  - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 458
  - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 458
  - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 458
  - activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 477
- ~ActiveMQMessage
  - activemq::commands::ActiveMQMessage, 461
- ~ActiveMQMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 469
  - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 473
  - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 481
  - activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 488
  - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 504
  - activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 509
- ~ActiveMQMessageTemplate
  - activemq::commands::ActiveMQMessageTemplate, 398
- ~ActiveMQObjectMessage
  - activemq::commands::ActiveMQObjectMessage, 414
- ~ActiveMQObjectMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 422
  - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 434
  - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 417
  - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 426
  - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 430
  - activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 430
- ~ActiveMQQueue
  - activemq::core::ActiveMQQueue, 454
- ~ActiveMQQueueBrowser
  - activemq::core::ActiveMQQueueBrowser, 458
- ~ActiveMQQueueMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 465
  - activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 465
  - activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 469
  - activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 469
  - activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 473
  - activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller, 473
- ~ActiveMQSession
  - activemq::core::ActiveMQSession, 488
- ~ActiveMQSessionExecutor
  - activemq::core::ActiveMQSessionExecutor, 504
- ~ActiveMQStreamMessage
  - activemq::commands::ActiveMQStreamMessage, 509
- ~ActiveMQStreamMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 528
  - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 528
  - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 532
  - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 532
  - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 532
  - activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 532
- ~ActiveMQTempDestination
  - activemq::core::ActiveMQTempDestination, 548
- ~ActiveMQTempDestinationMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 548

activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 556  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 567  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 567  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 552  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 559  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 563  
 activemq::wireformat::openwire::marshal::v7::ActiveMQTempDestinationMarshaller, 571  
 ~ActiveMQTempQueue, 661  
 activemq::commands::ActiveMQTempQueue, 661  
 ~ActiveMQTempQueueMarshaller, 673  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 583  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 595  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 579  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 587  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 591  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 599  
 ~ActiveMQTempTopic, 689  
 activemq::commands::ActiveMQTempTopic, 603  
 ~ActiveMQTempTopicMarshaller, 616  
 ~Appendable, 616  
 ~AprPool, 624  
 ~ArrayPointer, 608  
 ~AtomicBoolean, 612  
 ~AtomicInteger, 620  
 ~AtomicReference, 628  
 ~AtomicRefCounter, 714  
 ~AtomicReference, 714  
 ~BackupTransport, 645

- activemq::transport::failover::BackupTransportPool, 719
- ~BackupTransportPool
- activemq::transport::failover::BackupTransportPool, 721
- ~BaseCommand
- activemq::commands::BaseCommand, 724
- ~BaseCommandMarshaller
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 744
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 765
- activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 731
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 738
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 751
- activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 758
- ~BaseDataStreamMarshaller
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 776
- ~BaseDataStructure
- activemq::commands::BaseDataStructure, 794
- ~BindException
- decaf::net::BindException, 799
- ~BlockingByteArrayInputStream
- decaf::io::BlockingByteArrayInputStream, 802
- ~BlockingQueue
- decaf::util::concurrent::BlockingQueue, 807
- ~Boolean
- decaf::lang::Boolean, 812
- ~BooleanExpression
- activemq::commands::BooleanExpression, 816
- ~BooleanStream
- activemq::wireformat::openwire::utils::BooleanStream, 819
- ~BrokenBarrierException
- decaf::util::concurrent::BrokenBarrierException, 822
- ~BrokerError
- activemq::commands::BrokerError, 824
- ~BrokerException
- activemq::exceptions::BrokerException, 828
- ~BrokerId
- activemq::commands::BrokerId, 830
- ~BrokerIdMarshaller
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 841
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 853
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 833
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 837
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 845
- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 849
- ~BrokerInfo
- activemq::commands::BrokerInfo, 858
- ~BrokerInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 872
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 884
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 863
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 868
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 876
- activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 880
- ~Buffer
- decaf::nio::Buffer, 889
- ~BufferFactory
- decaf::internal::nio::BufferFactory, 903
- ~BufferOverflowException
- decaf::nio::BufferOverflowException, 915
- ~BufferUnderflowException
- decaf::nio::BufferUnderflowException, 918
- ~BufferedInputStream
- decaf::io::BufferedInputStream, 896
- ~BufferedOutputStream
- decaf::io::BufferedOutputStream, 900
- ~Byte
- decaf::lang::Byte, 920
- ~ByteArrayAdapter
- decaf::internal::util::ByteArrayAdapter, 935

- ~ByteBuffer
  - decaf::internal::nio::ByteBuffer, 966
- ~ByteArrayInputStream
  - decaf::io::ByteArrayInputStream, 988
- ~ByteArrayOutputStream
  - decaf::io::ByteArrayOutputStream, 993
- ~ByteBuffer
  - decaf::nio::ByteBuffer, 1000
- ~BytesMessage
  - cms::BytesMessage, 1026
- ~CMSException
  - cms::CMSException, 1132
- ~CMSExceptionSupport
  - activemq::util::CMSExceptionSupport, 1134
- ~CMSProperties
  - cms::CMSProperties, 1136
- ~CMSSecurityException
  - cms::CMSSecurityException, 1139
- ~CRC32
  - decaf::util::zip::CRC32, 1491
- ~CachedConsumer
  - activemq::cmsutil::CachedConsumer, 1042
- ~CachedProducer
  - activemq::cmsutil::CachedProducer, 1046
- ~Callable
  - decaf::util::concurrent::Callable, 1052
- ~CancellationException
  - decaf::util::concurrent::CancellationException, 1054
- ~Certificate
  - decaf::security::cert::Certificate, 1056
- ~CertificateEncodingException
  - decaf::security::cert::CertificateEncodingException, 1060
- ~CertificateException
  - decaf::security::cert::CertificateException, 1062
- ~CertificateExpiredException
  - decaf::security::cert::CertificateExpiredException, 1064
- ~CertificateNotYetValidException
  - decaf::security::cert::CertificateNotYetValidException, 1066
- ~CertificateParsingException
  - decaf::security::cert::CertificateParsingException, 1068
- ~CharArrayBuffer
  - decaf::internal::nio::CharArrayBuffer, 1083
- ~CharBuffer
  - decaf::nio::CharBuffer, 1092
- ~CharSequence
  - decaf::lang::CharSequence, 1108
- ~CheckedInputStream
  - decaf::util::zip::CheckedInputStream, 1111
- ~CheckedOutputStream
  - decaf::util::zip::CheckedOutputStream, 1113
- ~Checksum
  - decaf::util::zip::Checksum, 1115
- ~ClassCastException
  - decaf::lang::exceptions::ClassCastException, 1119
- ~CloseTransportsTask
  - activemq::transport::failover::CloseTransportsTask, 1122
- ~Closeable
  - cms::Closeable, 1120
  - decaf::io::Closeable, 1121
- ~CmsAccessor
  - activemq::cmsutil::CmsAccessor, 1124
- ~CmsDestinationAccessor
  - activemq::cmsutil::CmsDestinationAccessor, 1128
- ~CmsTemplate
  - activemq::cmsutil::CmsTemplate, 1143
- ~Collection
  - decaf::util::Collection, 1156
- ~Command
  - activemq::commands::Command, 1166
- ~CommandVisitor
  - activemq::state::CommandVisitor, 1173
- ~CommandVisitorAdapter
  - activemq::state::CommandVisitorAdapter, 1181
- ~Comparable
  - decaf::lang::Comparable, 1187
- ~Comparator
  - decaf::util::Comparator, 1190
- ~CompositeData
  - activemq::util::CompositeData, 1192
- ~CompositeTask
  - activemq::threads::CompositeTask, 1193
- ~CompositeTaskRunner
  - activemq::threads::CompositeTaskRunner, 1195
- ~CompositeTransport
  - activemq::transport::CompositeTransport, 1197



- ~ConcurrentMap
  - decaf::util::concurrent::ConcurrentMap, 1199
- ~ConcurrentStlMap
  - decaf::util::concurrent::ConcurrentStlMap, 1207
- ~Condition
  - decaf::util::concurrent::locks::Condition, 1222
- ~ConditionHandle
  - decaf::util::concurrent::ConditionHandle, 1227
- ~ConnectException
  - decaf::net::ConnectException, 1232
- ~Connection
  - cms::Connection, 1234
- ~ConnectionControl
  - activemq::commands::ConnectionControl, 1238
- ~ConnectionControlMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1251
  - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1263
  - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1243
  - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1247
  - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1255
  - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1259
- ~ConnectionError
  - activemq::commands::ConnectionError, 1267
- ~ConnectionErrorMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1283
  - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1271
  - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1275
  - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1279
  - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1287
  - activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller, 1291
- ~ConnectionFactory
  - cms::ConnectionFactory, 1295
- ~ConnectionId
  - activemq::commands::ConnectionId, 1298
- ~ConnectionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1314
  - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1302
  - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1306
  - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1310
  - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1318
  - activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1322
- ~ConnectionInfo
  - activemq::commands::ConnectionInfo, 1326
- ~ConnectionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1340
  - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1330
  - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1336
  - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1340
  - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1340
  - activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1350
- ~ConnectionMetaData
  - cms::ConnectionMetaData, 1356
- ~ConnectionState
  - activemq::state::ConnectionState, 1359
- ~ConnectionStateTracker
  - activemq::state::ConnectionStateTracker, 1362
- ~ConsoleHandler
  - decaf::util::logging::ConsoleHandler, 1368
- ~ConsumerControl
  - activemq::commands::ConsumerControl, 1370
- ~ConsumerControlMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1387
  - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1374
  - activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1379

activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1383  
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1472  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1391  
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1480  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1395  
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1484  
 ~ConsumerId  
 ~CountDownLatch  
 activemq::commands::ConsumerId, 1399  
 decaf::util::concurrent::CountDownLatch, 1487  
 ~ConsumerIdMarshaller  
 ~DataArrayResponse  
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1415  
 activemq::commands::DataArrayResponse, 1494  
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1403  
 ~DataArrayResponseMarshaller  
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1407  
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1509  
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1497  
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1501  
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1505  
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1517  
 ~ConsumerInfo  
 activemq::commands::ConsumerInfo, 1428  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1517  
 ~ConsumerInfoMarshaller  
 ~DataFormatException  
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1448  
 decaf::util::zip::DataFormatException, 1522  
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1435  
 ~DataInput  
 decaf::io::DataInput, 1524  
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1440  
 ~DataInputStream  
 decaf::io::DataInputStream, 1534  
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1444  
 ~DataOutput  
 decaf::io::DataOutput, 1542  
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1452  
 ~DataOutputStream  
 decaf::io::DataOutputStream, 1548  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1456  
 ~DataResponse  
 activemq::commands::DataResponse, 1551  
 ~ConsumerState  
 activemq::state::ConsumerState, 1459  
 ~DataResponseMarshaller  
 ~ControlCommand  
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1574  
 activemq::commands::ControlCommand, 1460  
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1562  
 ~ControlCommandMarshaller  
 ~ControlCommandMarshaller  
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1476  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1566  
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1463  
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1570  
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1468  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1554

- activemq::wireformat::openwire::marshal::DestinationResponseMarshaller, 1558
- activemq::commands::DestinationInfo, 1693
- ~DataStreamMarshaller
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1578
- ~DataStructure
- activemq::commands::DataStructure, 1628
- ~Date
- decaf::util::Date, 1634
- ~DecafRuntime
- decaf::internal::DecafRuntime, 1638
- ~DedicatedTaskRunner
- activemq::threads::DedicatedTaskRunner, 1639
- ~DefaultPrefetchPolicy
- activemq::core::policies::DefaultPrefetchPolicy, 1641
- ~DefaultRedeliveryPolicy
- activemq::core::policies::DefaultRedeliveryPolicy, 1645
- ~DefaultSSLContext
- decaf::internal::net::ssl::DefaultSSLContext, 1657
- ~DefaultSSLServerSocketFactory
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1660
- ~DefaultSSLSocketFactory
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1666
- ~DefaultServerSocketFactory
- decaf::internal::net::DefaultServerSocketFactory, 1650
- ~DefaultSocketFactory
- decaf::internal::net::DefaultSocketFactory, 1654
- ~DefaultTransportListener
- activemq::transport::DefaultTransportListener, 1671
- ~Deflater
- decaf::util::zip::Deflater, 1674
- ~DeflaterOutputStream
- decaf::util::zip::DeflaterOutputStream, 1684
- ~Delayed
- decaf::util::concurrent::Delayed, 1687
- ~DeliveryMode
- cms::DeliveryMode, 1688
- ~Destination
- cms::Destination, 1690
- ~DestinationInfoMarshaller
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1709
- activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1697
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1701
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1705
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1717
- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1713
- ~DestinationResolver
- activemq::cmsutil::DestinationResolver, 1720
- ~DiscoveryEvent
- activemq::commands::DiscoveryEvent, 1723
- ~DiscoveryEventMarshaller
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1742
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1730
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1734
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1738
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1746
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1726
- ~Dispatcher
- activemq::core::Dispatcher, 1750
- ~Double
- decaf::lang::Double, 1753
- ~DoubleArrayBuffer
- decaf::internal::nio::DoubleArrayBuffer, 1768
- ~DoubleBuffer
- decaf::nio::DoubleBuffer, 1776
- ~DynamicDestinationResolver
- activemq::cmsutil::DynamicDestinationResolver, 1787
- ~EOFException
- decaf::io::EOFException, 1791
- ~Entry

- decaf::util::Map::Entry, 1789
- ~ErrorManager
  - decaf::util::logging::ErrorManager, 1793
- ~Exception
  - decaf::lang::Exception, 1796
- ~ExceptionListener
  - cms::ExceptionListener, 1801
- ~ExceptionResponse
  - activemq::commands::ExceptionResponse, 1802
- ~ExceptionResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1826
  - activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1810
  - activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1814
  - activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1822
  - activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1818
  - activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1805
- ~ExecutionException
  - decaf::util::concurrent::ExecutionException, 1831
- ~Executor
  - decaf::util::concurrent::Executor, 1833
- ~ExecutorService
  - decaf::util::concurrent::ExecutorService, 1834
- ~FailoverTransport
  - activemq::transport::failover::FailoverTransport, 1837
- ~FailoverTransportFactory
  - activemq::transport::failover::FailoverTransportFactory, 1847
- ~FailoverTransportListener
  - activemq::transport::failover::FailoverTransportListener, 1849
- ~FileDescriptor
  - decaf::io::FileDescriptor, 1852
- ~Filter
  - decaf::util::logging::Filter, 1853
- ~FilterInputStream
  - decaf::io::FilterInputStream, 1856
- ~FilterOutputStream
  - decaf::io::FilterOutputStream, 1862
- ~Float
  - decaf::lang::Float, 1867
- ~FloatArrayBuffer
  - decaf::internal::nio::FloatArrayBuffer, 1882
- ~FloatBuffer
  - decaf::nio::FloatBuffer, 1890
- ~FlushCommand
  - activemq::commands::FlushCommand, 1901
- ~FlushCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1920
  - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1908
  - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1912
  - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1916
  - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1924
  - activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 1904
- ~Flushable
  - ExceptionResponseMarshaller, 1906
- ~Formatter
  - decaf::util::logging::Formatter, 1928
- ~Future
  - decaf::util::concurrent::Future, 1930
- ~FutureResponse
  - activemq::transport::correlator::FutureResponse, 1933
- ~GeneralSecurityException
  - decaf::security::GeneralSecurityException, 1936
- ~GenericResource
  - decaf::internal::util::GenericResource, 1938
- ~Handler
  - decaf::util::logging::Handler, 1942
- ~HexStringParser
  - decaf::internal::util::HexStringParser, 1946
- ~HexTable
  - activemq::wireformat::openwire::utils::HexTable, 1947
- ~HttpRetryException
  - decaf::net::HttpRetryException, 1950
- ~IOException
  - decaf::io::IOException, 2104
- ~IOTransport
  - activemq::transport::IOTransport, 2107
- ~IdGenerator
  - activemq::util::IdGenerator, 1951

- ~IllegalArgumentException
  - decaf::lang::exceptions::IllegalArgumentException, 2066
  - 1955
- ~IllegalMonitorStateException
  - decaf::lang::exceptions::IllegalMonitorStateException, 2070
  - 1957
- ~IllegalStateException
  - cms::IllegalStateException, 1959
  - decaf::lang::exceptions::IllegalStateException, 2058
  - 1961
- ~IllegalThreadStateException
  - decaf::lang::exceptions::IllegalThreadStateException, 2085
  - 1964
- ~InactivityMonitor
  - activemq::transport::inactivity::InactivityMonitor, 1965
- ~IndexOutOfBoundsException
  - decaf::lang::exceptions::IndexOutOfBoundsException, 1969
- ~Inet4Address
  - decaf::net::Inet4Address, 1971
- ~Inet6Address
  - decaf::net::Inet6Address, 1974
- ~InetAddress
  - decaf::net::InetAddress, 1976
- ~InetSocketAddress
  - decaf::net::InetSocketAddress, 1982
- ~Inflater
  - decaf::util::zip::Inflater, 1987
- ~InflaterInputStream
  - decaf::util::zip::InflaterInputStream, 1998
- ~InputStream
  - decaf::io::InputStream, 2004
- ~InputStreamReader
  - decaf::io::InputStreamReader, 2014
- ~IntArrayBuffer
  - decaf::internal::nio::IntArrayBuffer, 2021
- ~IntBuffer
  - decaf::nio::IntBuffer, 2029
- ~Integer
  - decaf::lang::Integer, 2041
- ~IntegerResponse
  - activemq::commands::IntegerResponse, 2055
- ~IntegerResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2074
  - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2062
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2066
- activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2070
- activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2078
- activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2058
- InternalCommandListener
  - activemq::transport::mock::InternalCommandListener, 2085
- ~InterruptedException
  - decaf::lang::exceptions::InterruptedException, 2088
- ~InterruptedIOException
  - decaf::io::InterruptedIOException, 2091
- ~InvalidClientIdException
  - cms::InvalidClientIdException, 2092
- ~InvalidDestinationException
  - cms::InvalidDestinationException, 2093
- ~InvalidKeyException
  - decaf::security::InvalidKeyException, 2096
- ~InvalidMarkException
  - decaf::nio::InvalidMarkException, 2098
- ~InvalidSelectorException
  - cms::InvalidSelectorException, 2100
- ~InvalidStateException
  - decaf::lang::exceptions::InvalidStateException, 2102
- ~Iterable
  - decaf::lang::Iterable, 2113
- ~Iterator
  - decaf::util::Iterator, 2115
- ~JournalQueueAck
  - activemq::commands::JournalQueueAck, 2117
- ~JournalQueueAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2140
  - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2124
  - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2132
  - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2136
  - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2148
  - activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2140
- ~JournalTopicAck

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2315
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2444
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2319
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2327
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2323
- activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2311
- ~Lock
  - decaf::util::concurrent::Lock, 2335
  - decaf::util::concurrent::locks::Lock, 2338
- ~LockSupport
  - decaf::util::concurrent::locks::LockSupport, 2343
- ~LogManager
  - decaf::util::logging::LogManager, 2366
- ~LogRecord
  - decaf::util::logging::LogRecord, 2371
- ~LogWriter
  - decaf::util::logging::LogWriter, 2376
- ~Logger
  - decaf::util::logging::Logger, 2348
- ~LoggerHierarchy
  - decaf::util::logging::LoggerHierarchy, 2357
- ~LoggingInputStream
  - activemq::io::LoggingInputStream, 2358
- ~LoggingOutputStream
  - activemq::io::LoggingOutputStream, 2360
- ~LoggingTransport
  - activemq::transport::logging::LoggingTransport, 2361
- ~Long
  - decaf::lang::Long, 2380
- ~LongArrayBuffer
  - decaf::internal::nio::LongArrayBuffer, 2397
- ~LongBuffer
  - decaf::nio::LongBuffer, 2406
- ~LongSequenceGenerator
  - activemq::util::LongSequenceGenerator, 2416
- ~MalformedURLException
  - decaf::net::MalformedURLException, 2418
- ~Map
  - decaf::util::Map, 2420
- ~MapMessage
  - cms::MapMessage, 2434
- ~MarkBlockLogger
  - decaf::util::logging::MarkBlockLogger, 2444
- ~MarshallingSupport
  - activemq::util::MarshallingSupport, 2452
- ~Math
  - decaf::lang::Math, 2457
- ~MemoryUsage
  - activemq::util::MemoryUsage, 2473
- ~Message
  - activemq::commands::Message, 2480
  - cms::Message, 2497
- ~MessageAck
  - activemq::commands::MessageAck, 2522
- ~MessageAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2543
  - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2531
  - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2535
  - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2539
  - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2547
  - activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2527
- ~MessageConsumer
  - cms::MessageConsumer, 2551
- ~MessageCreator
  - activemq::cmsutil::MessageCreator, 2554
- ~MessageDispatch
  - activemq::commands::MessageDispatch, 2556
- ~MessageDispatchChannel

activemq::core::MessageDispatchChannel, 2561  
 ~MessageDispatchMarshaller 2583  
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2583  
 ~MessageMarshaller 2583  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2567  
 ~MessageMarshaller 2671  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2571  
 ~MessageMarshaller 2662  
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2579  
 ~MessageMarshaller 2658  
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2575  
 ~MessageMarshaller 2667  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2587  
 ~MessageMarshaller 2654  
 ~MessageDispatchNotification 2592  
 activemq::commands::MessageDispatchNotification, 2592  
 ~MessageNotReadableException 2592  
 ~MessageDispatchNotificationMarshaller 2612  
 cms::MessageNotReadableException, 2612  
 ~MessageNotWriteableException 2612  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2600  
 ~MessageNotWriteableException 2681  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2604  
 cms::MessageProducer, 2683  
 ~MessageProducer 2608  
 ~MessageDispatchNotificationMarshaller, 2608  
 activemq::wireformat::openwire::utils::MessagePropertyInterface, 2617  
 ~MessagePull 2617  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2596  
 ~MessagePullMarshaller 2696  
 ~MessageEOFException 2717  
 cms::MessageEOFException, 2622  
 ~MessageEnumeration 2701  
 cms::MessageEnumeration, 2620  
 ~MessageFormatException 2709  
 cms::MessageFormatException, 2623  
 ~MessageId 2713  
 activemq::commands::MessageId, 2625  
 ~MessageIdMarshaller 2705  
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2649  
 ~MessageIdMarshaller, 2629  
 ~MockTransport 2726  
 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2641  
 ~MockTransport, 2726  
 ~MockMessageIdMarshaller, 2633  
 activemq::transport::mock::MockTransportFactory, 2637  
 ~Mutex 2637



- decaf::util::concurrent::Mutex, 2737
- ~MutexHandle
  - decaf::util::concurrent::MutexHandle, 274
- ~Network
  - decaf::internal::net::Network, 2745
- ~NetworkBridgeFilter
  - activemq::commands::NetworkBridgeFilter, 2747
- ~NetworkBridgeFilterMarshaller
  - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2770
  - activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2750
  - activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2762
  - activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2766
  - activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2758
  - activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2754
- ~NoRouteToHostException
  - decaf::net::NoRouteToHostException, 2775
- ~NoSuchAlgorithmException
  - decaf::security::NoSuchAlgorithmException, 2778
- ~NoSuchElementException
  - decaf::lang::exceptions::NoSuchElementException, 2780
- ~NoSuchProviderException
  - decaf::security::NoSuchProviderException, 2783
- ~NullPointerException
  - decaf::lang::exceptions::NullPointerException, 2785
- ~Number
  - decaf::lang::Number, 2787
- ~NumberFormatException
  - decaf::lang::exceptions::NumberFormatException, 2791
- ~ObjectMessage
  - cms::ObjectMessage, 2792
- ~OpenSSLContextSpi
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2793
- ~OpenSSLParameters
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796
- ~OpenSSLServerSocket
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2799
  - OpenSSLServerSocketFactory
    - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2805
  - ~OpenSSLSocket
    - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2813
  - ~OpenSSLSocketException
    - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2824
  - ~OpenSSLSocketFactory
    - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2828
  - ~OpenSSLSocketInputStream
    - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2833
  - ~OpenSSLSocketOutputStream
    - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2838
  - ~OpenWireFormat
    - activemq::wireformat::openwire::OpenWireFormat, 2840
  - ~OpenWireFormatFactory
    - activemq::wireformat::openwire::OpenWireFormatFactory, 2850
  - ~OpenWireFormatNegotiator
    - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2852
  - ~OpenWireResponseBuilder
    - activemq::wireformat::openwire::OpenWireResponseBuilder, 2855
  - ~OutputStream
    - decaf::io::OutputStream, 2858
  - ~OutputStreamWriter
    - decaf::io::OutputStreamWriter, 2865
  - ~PartialCommand
    - activemq::commands::PartialCommand, 2867
  - ~PartialCommandMarshaller
    - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2892
    - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2875
    - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2884
    - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2888
    - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2879

- activemq::wireformat::openwire::marshal::PortUnreachableException, 2871
- ~Pointer
  - decaf::lang::Pointer, 2900
- ~PooledSession
  - activemq::cmsutil::PooledSession, 2907
- ~PooledThread
  - decaf::util::concurrent::PooledThread, 2919
- ~PooledThreadListener
  - decaf::util::concurrent::PooledThreadListener, 2921
- ~PortUnreachableException
  - decaf::net::PortUnreachableException, 2924
- ~PrefetchPolicy
  - activemq::core::PrefetchPolicy, 2926
- ~PrimitiveList
  - activemq::util::PrimitiveList, 2931
- ~PrimitiveMap
  - activemq::util::PrimitiveMap, 2943
- ~PrimitiveTypesMarshaller
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2952
- ~PrimitiveValueConverter
  - activemq::util::PrimitiveValueConverter, 2959
- ~PrimitiveValueNode
  - activemq::util::PrimitiveValueNode, 2967
- ~Principal
  - decaf::security::Principal, 2975
- ~PriorityQueue
  - decaf::util::PriorityQueue, 2979
- ~ProducerAck
  - activemq::commands::ProducerAck, 2985
- ~ProducerAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 3009
  - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2989
  - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2997
  - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2993
  - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 3001
  - activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3005
- ~ProducerCallback
  - activemq::cmsutil::ProducerCallback, 3012
- ~ProducerCommandMarshaller
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3013
- ~ProducerId
  - activemq::commands::ProducerId, 3016
- ~ProducerIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3040
  - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3020
  - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3028
  - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3024
  - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3032
  - activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3036
- ~ProducerInfo
  - activemq::commands::ProducerInfo, 3044
- ~ProducerInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3052
  - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3053
  - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3065
  - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3048
  - activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3061
  - activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3069
- ~ProducerState
  - activemq::state::ProducerState, 3072
- ~Properties
  - decaf::util::Properties, 3074
- ~PropertiesChangeListener
  - decaf::util::PropertiesChangeListener, 3083
- ~ProtocolExceptionMarshaller
  - decaf::net::ProtocolException, 3085
- ~PublicKey
  - decaf::security::PublicKey, 3086
- ~PushbackInputStreamMarshaller
  - decaf::io::PushbackInputStream, 3089
- ~Queue
  - cms::Queue, 3094
  - decaf::util::Queue, 3095
- QueueBrowser

- cms::QueueBrowser, 3099
- ~ReadChecker
  - activemq::transport::inactivity::ReadChecker, 3179
  - 3108
- ~ReadOnlyBufferException
  - decaf::nio::ReadOnlyBufferException, 3175
  - 3116
- ~ReadWriteLock
  - decaf::util::concurrent::locks::ReadWriteLock, 3191
  - 3118
- ~Readable
  - decaf::lang::Readable, 3187
  - 3106
- ~Reader
  - decaf::io::Reader, 3183
  - 3110
- ~ReceiveExecutor
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3195
  - 3120
- ~RedeliveryPolicy
  - activemq::core::RedeliveryPolicy, 3202
  - 3123
- ~ReentrantLock
  - decaf::util::concurrent::locks::ReentrantLock, 3206
  - 3128
- ~RejectedExecutionException
  - decaf::util::concurrent::RejectedExecutionException, 3210
  - 3136
- ~RejectedExecutionHandler
  - decaf::util::concurrent::RejectedExecutionHandler, 3218
  - 3137
- ~RemoveInfo
  - activemq::commands::RemoveInfo, 3214
  - 3138
- ~RemoveInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3221
  - 3154
- ~Resource
  - decaf::internal::util::Resource, 3222
  - 3142
- ~ResourceLifecycleManager
  - activemq::cmsutil::ResourceLifecycleManager, 3223
  - 3150
- ~Response
  - decaf::internal::util::ResourceLifecycleManager, 3224
  - 3162
- ~ResponseBuilder
  - activemq::wireformat::openwire::marshal::v5::ResponseBuilder, 3228
  - 3158
- ~ResponseCorrelator
  - activemq::transport::correlator::ResponseCorrelator, 3232
  - 3146
- ~RemoveSubscriptionInfo
  - activemq::commands::RemoveSubscriptionInfo, 3234
  - 3166
- ~RemoveSubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3256
  - 3170
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3179
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3175
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3191
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3187
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3183
- ~ReplayCommand
  - activemq::commands::ReplayCommand, 3195
- ~ReplayCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3202
  - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3206
  - activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3210
  - activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3198
  - activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3218
  - activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3214
- ~ResolveProducerExecutor
  - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3221
- ~ResolveReceiveExecutor
  - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3222
- ~RemoveInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3154
  - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3142
  - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3150
  - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3162
  - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3158
  - activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3146
- ~Response
  - activemq::commands::Response, 3228
- ~ResponseBuilder
  - activemq::transport::correlator::ResponseBuilder, 3232
- ~ResponseCorrelator
  - activemq::transport::correlator::ResponseCorrelator, 3234
- ~ResponseMarshaller
  - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3256

- activemq::wireformat::openwire::marshal::SessionCallbackMarshaller, 3242
- activemq::wireformat::openwire::marshal::SessionCallbackMarshaller, 3251
- activemq::wireformat::openwire::marshal::SessionIdMarshaller, 3237
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3247
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3261
- ~Runnable
  - decaf::lang::Runnable, 3265
- ~Runtime
  - decaf::lang::Runtime, 3266
- ~RuntimeException
  - decaf::lang::exceptions::RuntimeException, 3269
- ~SSLContext
  - decaf::net::ssl::SSLContext, 3490
- ~SSLContextSpi
  - decaf::net::ssl::SSLContextSpi, 3493
- ~SSLParameters
  - decaf::net::ssl::SSLParameters, 3496
- ~SSLServerSocket
  - decaf::net::ssl::SSLServerSocket, 3501
- ~SSLServerSocketFactory
  - decaf::net::ssl::SSLServerSocketFactory, 3505
- ~SSLSocket
  - decaf::net::ssl::SSLSocket, 3510
- ~SSLSocketFactory
  - decaf::net::ssl::SSLSocketFactory, 3516
- ~SecureRandom
  - decaf::security::SecureRandom, 3272
- ~SecureRandomImpl
  - decaf::internal::security::SecureRandomImpl, 3276
- ~SecureRandomSpi
  - decaf::security::SecureRandomSpi, 3279
- ~Semaphore
  - decaf::util::concurrent::Semaphore, 3283
- ~SendExecutor
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3291
- ~ServerSocket
  - decaf::net::ServerSocket, 3295
- ~ServerSocketFactory
  - decaf::net::ServerSocketFactory, 3302
- ~Session
  - cms::Session, 3308
- ~SessionCallback
  - activemq::cmsutil::SessionCallback, 3320
- ~SessionId
  - activemq::commands::SessionId, 3322
- ~SessionInfo
  - activemq::commands::SessionInfo, 3349
- ~SessionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3361
  - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3369
  - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3365
  - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3373
  - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3357
  - activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3353
- ~SessionPool
  - activemq::cmsutil::SessionPool, 3377
- ~SessionState
  - activemq::state::SessionState, 3378
- ~Set
  - decaf::util::Set, 3380
- ~Short
  - decaf::lang::Short, 3382
- ~ShortArrayBuffer
  - decaf::internal::nio::ShortArrayBuffer, 3395
- ~ShortBuffer
  - decaf::nio::ShortBuffer, 3403
- ~ShutdownInfo
  - activemq::commands::ShutdownInfo, 3414
- ~ShutdownInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3425

- activemq::wireformat::openwire::marshal::ShutdownInfoMarshaller, 3421
- activemq::wireformat::openwire::marshal::ShutdownInfoMarshaller, 3433
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3437
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 3429
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3417
- ~SignatureException
  - decaf::security::SignatureException, 3442
- ~SimpleFormatter
  - decaf::util::logging::SimpleFormatter, 3443
- ~SimpleLogger
  - decaf::util::logging::SimpleLogger, 3444
- ~Socket
  - decaf::net::Socket, 3451
- ~SocketAddress
  - decaf::net::SocketAddress, 3464
- ~SocketException
  - decaf::net::SocketException, 3466
- ~SocketFactory
  - decaf::net::SocketFactory, 3468
- ~SocketFileDescriptor
  - decaf::internal::net::SocketFileDescriptor, 3472
- ~SocketImpl
  - decaf::net::SocketImpl, 3474
- ~SocketImplFactory
  - decaf::net::SocketImplFactory, 3482
- ~SocketOptions
  - decaf::net::SocketOptions, 3483
- ~SocketTimeoutException
  - decaf::net::SocketTimeoutException, 3489
- ~SslTransport
  - activemq::transport::tcp::SslTransport, 3519
- ~SslTransportFactory
  - activemq::transport::tcp::SslTransportFactory, 3520
- ~StandardErrorOutputStream
  - decaf::internal::io::StandardErrorOutputStream, 3522
- ~StandardInputStream
  - decaf::internal::io::StandardInputStream, 3524
- ~StandardOutputStream
  - decaf::internal::io::StandardOutputStream, 3526
- ~StList
  - decaf::util::StList, 3528
- ~StMap
  - decaf::util::StMap, 3547
- ~StQueue
  - decaf::util::StQueue, 3558
- ~StSet
  - decaf::util::StSet, 3567
- ~StompFrame
  - activemq::wireformat::stomp::StompFrame, 3578
- ~StompHelper
  - activemq::wireformat::stomp::StompHelper, 3582
- ~StompWireFormat
  - activemq::wireformat::stomp::StompWireFormat, 3587
- ~StompWireFormatFactory
  - activemq::wireformat::stomp::StompWireFormatFactory, 3590
- ~Stoppable
  - cms::Stoppable, 3591
- ~StreamHandler
  - decaf::util::logging::StreamHandler, 3593
- ~StreamMessage
  - cms::StreamMessage, 3597
- ~String
  - decaf::lang::String, 3611
- ~StringTokenizer
  - decaf::util::StringTokenizer, 3614
- ~SubscriptionInfo
  - activemq::commands::SubscriptionInfo, 3617
- SubscriptionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3625
  - activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3641
  - activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3621
  - activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3633
  - activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 3629

- activemq::wireformat::openwire::marshal::Throwable, 3637
- ~Synchronizable
  - decaf::util::concurrent::Synchronizable, 3645
- ~SynchronizableImpl
  - decaf::internal::util::concurrent::SynchronizableImpl, 3656
- ~Synchronization
  - activemq::core::Synchronization, 3659
- ~SynchronousQueue
  - decaf::util::concurrent::SynchronousQueue, 3662
- ~System
  - decaf::lang::System, 3672
- ~Task
  - activemq::threads::Task, 3679
- ~TaskListener
  - decaf::util::concurrent::TaskListener, 3679
- ~TaskRunner
  - activemq::threads::TaskRunner, 3681
- ~TcpSocket
  - decaf::internal::net::tcp::TcpSocket, 3685
- ~TcpSocketInputStream
  - decaf::internal::net::tcp::TcpSocketInputStream, 3692
- ~TcpSocketOutputStream
  - decaf::internal::net::tcp::TcpSocketOutputStream, 3695
- ~TcpTransport
  - activemq::transport::tcp::TcpTransport, 3697
- ~TcpTransportFactory
  - activemq::transport::tcp::TcpTransportFactory, 3700
- ~TemporaryQueue
  - cms::TemporaryQueue, 3702
- ~TemporaryTopic
  - cms::TemporaryTopic, 3704
- ~TextMessage
  - cms::TextMessage, 3705
- ~Thread
  - decaf::lang::Thread, 3711
- ~ThreadFactory
  - decaf::util::concurrent::ThreadFactory, 3717
- ~ThreadGroup
  - decaf::lang::ThreadGroup, 3718
- ~ThreadPool
  - decaf::util::concurrent::ThreadPool, 3720
- ~Throwable
  - decaf::lang::Throwable, 3725
- ~TimeUnit
  - decaf::util::concurrent::TimeUnit, 3750
- ~TimeoutException
  - decaf::util::concurrent::TimeoutException, 3730
- ~Timer
  - decaf::util::Timer, 3733
- ~TimerTask
  - decaf::util::TimerTask, 3743
- ~TimerTaskHeap
  - decaf::internal::util::TimerTaskHeap, 3746
- ~Topic
  - cms::Topic, 3758
- ~Tracked
  - activemq::state::Tracked, 3759
- ~TransactionId
  - activemq::commands::TransactionId, 3760
- ~TransactionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3767
  - activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3771
  - activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3775
  - activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3779
  - activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3764
  - activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3782
- ~TransactionInfo
  - activemq::commands::TransactionInfo, 3786
- ~TransactionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3794
  - activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3810
  - activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3798
  - activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3806
  - activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3790
  - activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3802
- ~TransactionState
  - activemq::state::TransactionState, 3814

- ~TransferQueue
  - decaf::internal::util::concurrent::TransferQueue, 3816
- ~TransferStack
  - decaf::internal::util::concurrent::TransferStack, 3818
- ~Transport
  - activemq::transport::Transport, 3820
- ~TransportFactory
  - activemq::transport::TransportFactory, 3826
- ~TransportFilter
  - activemq::transport::TransportFilter, 3829
- ~TransportListener
  - activemq::transport::TransportListener, 3836
- ~TransportRegistry
  - activemq::transport::TransportRegistry, 3838
- ~URI
  - decaf::net::URI, 3857
- ~URIEncoderDecoder
  - decaf::internal::net::URIEncoderDecoder, 3866
- ~URIHelper
  - decaf::internal::net::URIHelper, 3869
- ~URIPool
  - activemq::transport::failover::URIPool, 3875
- ~URISyntaxException
  - decaf::net::URISyntaxException, 3883
- ~URIType
  - decaf::internal::net::URIType, 3885
- ~URL
  - decaf::net::URL, 3893
- ~URLDecoder
  - decaf::net::URLDecoder, 3894
- ~URLEncoder
  - decaf::net::URLEncoder, 3894
- ~UTFDataFormatException
  - decaf::io::UTFDataFormatException, 3899
- ~UUID
  - decaf::util::UUID, 3902
- ~UncaughtExceptionHandler
  - decaf::lang::Thread::UncaughtExceptionHandler, 3841
- ~UnknownHostException
  - decaf::net::UnknownHostException, 3843
- ~UnknownServiceException
  - decaf::net::UnknownServiceException, 3846
- ~UnsupportedEncodingException
  - decaf::io::UnsupportedEncodingException, 3849
- ~UnsupportedOperationException
  - cms::UnsupportedOperationException, 3853
  - decaf::lang::exceptions::UnsupportedOperationException, 3851
- ~Usage
  - activemq::util::Usage, 3896
- WireFormat
  - activemq::wireformat::WireFormat, 3908
- ~WireFormatFactory
  - activemq::wireformat::WireFormatFactory, 3911
- ~WireFormatInfo
  - activemq::commands::WireFormatInfo, 3914
- ~WireFormatInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3940
  - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3932
  - activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3944
  - activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3936
  - activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3924
  - activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3928
- ~WireFormatNegotiator
  - activemq::wireformat::WireFormatNegotiator, 3947
- ~WireFormatRegistry
  - activemq::wireformat::WireFormatRegistry, 3948
- ~WriteChecker
  - activemq::transport::inactivity::WriteChecker, 3951
- ~Writer
  - decaf::io::Writer, 3952
- ~X500Principal
  - decaf::security::auth::x500::X500Principal, 3957
- ~X509Certificate
  - decaf::security::cert::X509Certificate, 3959

- ~XATransactionId
  - activemq::commands::XATransactionId, 3962
- ~XATransactionIdMarshaller
  - activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 3977
  - activemq::wireformat::openwire::marshaller::XATransactionIdMarshaller, 3969
  - activemq::wireformat::openwire::marshaller::v3::XATransactionIdMarshaller, 3981
  - activemq::wireformat::openwire::marshaller::v4::XATransactionIdMarshaller, 3973
  - activemq::wireformat::openwire::marshaller::v5::XATransactionIdMarshaller, 3985
  - activemq::wireformat::openwire::marshaller::v6::XATransactionIdMarshaller, 3965
- ~XMLFormatter
  - decaf::util::logging::XMLFormatter, 3989
- ~ZipException
  - decaf::util::zip::ZipException, 3993
- \_FALSE
  - decaf::lang::Boolean, 815
- \_TRUE
  - decaf::lang::Boolean, 815
- \_array
  - decaf::internal::nio::CharArrayBuffer, 1089
- \_capacity
  - decaf::nio::Buffer, 893
- \_dist\_code
  - deflate.h, 4421
  - trees.h, 4426
- \_length\_code
  - deflate.h, 4421
  - trees.h, 4427
- \_limit
  - decaf::nio::Buffer, 893
- \_mark
  - decaf::nio::Buffer, 893
- \_markSet
  - decaf::nio::Buffer, 893
- \_position
  - decaf::nio::Buffer, 893
- \_tr\_tally\_dist
  - deflate.h, 4419
- \_tr\_tally\_lit
  - deflate.h, 4419
- ABORT
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- abs
  - decaf::lang::Math, 2457, 2458
- AbstractCollection
  - decaf::util::AbstractCollection, 149
- AbstractQueue
  - decaf::util::AbstractQueue, 164
- accept
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2799
- accept
  - decaf::internal::net::tcp::TcpSocket, 3685
- accept
  - decaf::net::SocketImpl, 3474
- accept
  - decaf::net::Socket, 3451
- ACK
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- ACK\_AUTO
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- ACK\_CLIENT
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- ACK\_INDIVIDUAL
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- ACK\_TYPE\_CONSUMED
  - activemq::core::ActiveMQConstants, 280
- ACK\_TYPE\_DELIVERED
  - activemq::core::ActiveMQConstants, 280
- ACK\_TYPE\_INDIVIDUAL
  - activemq::core::ActiveMQConstants, 280
- ACK\_TYPE\_POISON
  - activemq::core::ActiveMQConstants, 280
- ACK\_TYPE\_REDELIVERED
  - activemq::core::ActiveMQConstants, 280
- acknowledge
  - activemq::commands::ActiveMQMessageTemplate, 398
  - activemq::core::ActiveMQConsumer, 284, 285
  - activemq::core::ActiveMQSession, 488
  - cms::Message, 2497
- acknowledgeMessage
  - activemq::core::ActiveMQAckHandler, 172
- AcknowledgeMode
  - cms::Session, 3308
- ActiveMQ
  - activemq::core::ActiveMQConstants, 280



- ackType
  - activemq::commands::MessageAck, 2526
- acquire
  - decaf::util::concurrent::Semaphore, 3283, 3284
- acquireUninterruptibly
  - decaf::util::concurrent::Semaphore, 3284, 3285
- action
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3014
- activemq, 93
- activemq/exceptions/ExceptionDefines.h
  - AMQ\_CATCH\_EXCEPTION\_CONVERT, 4052
  - AMQ\_CATCH\_NOTHROW, 4052
  - AMQ\_CATCH\_RETHROW, 4052
  - AMQ\_CATCHALL\_NOTHROW, 4053
  - AMQ\_CATCHALL\_THROW, 4053
- activemq/util/Config.h
  - AMQCPP\_API, 4086
  - HAVE\_PTHREAD\_H, 4086
  - HAVE\_UUID\_T, 4086
  - HAVE\_UUID\_UUID\_H, 4086
- activemq::cmsutil, 94
- activemq::cmsutil::CachedConsumer, 1041
  - ~CachedConsumer, 1042
  - CachedConsumer, 1042
  - close, 1042
  - getMessageListener, 1042
  - getMessageSelector, 1042
  - operator=, 1043
  - receive, 1043
  - receiveNoWait, 1043
  - setMessageListener, 1044
- activemq::cmsutil::CachedProducer, 1044
  - ~CachedProducer, 1046
  - CachedProducer, 1045
  - close, 1046
  - getDeliveryMode, 1046
  - getDisableMessageID, 1046
  - getDisableMessageTimeStamp, 1046
  - getPriority, 1047
  - getTimeToLive, 1047
  - operator=, 1047
  - send, 1047–1049
  - setDeliveryMode, 1050
  - setDisableMessageID, 1050
  - setDisableMessageTimeStamp, 1050
  - setPriority, 1051
  - setTimeToLive, 1051
- activemq::cmsutil::CmsAccessor, 1123
  - ~CmsAccessor, 1124
  - checkConnectionFactory, 1124
  - CmsAccessor, 1124
  - createConnection, 1125
  - createSession, 1125
  - destroy, 1125
  - getConnectionFactory, 1126
  - getResourceLifecycleManager, 1126
  - getSessionAcknowledgeMode, 1126
  - init, 1126
  - operator=, 1127
  - setConnectionFactory, 1127
  - setSessionAcknowledgeMode, 1127
- activemq::cmsutil::CmsDestinationAccessor, 1127
  - ~CmsDestinationAccessor, 1128
  - checkDestinationResolver, 1128
  - CmsDestinationAccessor, 1128
  - destroy, 1129
  - getDestinationResolver, 1129
  - init, 1129
  - isPubSubDomain, 1129
  - operator=, 1129
  - resolveDestinationName, 1129
  - setDestinationResolver, 1130
  - setPubSubDomain, 1130
- activemq::cmsutil::CmsTemplate, 1140
  - ~CmsTemplate, 1143
  - CmsTemplate, 1143
  - DEFAULT\_PRIORITY, 1153
  - DEFAULT\_TIME\_TO\_LIVE, 1153
  - destroy, 1143
  - execute, 1144, 1145
  - getDefaultDestination, 1145
  - getDefaultDestinationName, 1145
  - getDeliveryMode, 1146
  - getPriority, 1146
  - getReceiveTimeout, 1146
  - getTimeToLive, 1146
  - init, 1146
  - isExplicitQosEnabled, 1146
  - isMessageIdEnabled, 1147
  - isMessageTimestampEnabled, 1147
  - isNoLocal, 1147
  - operator=, 1147
  - ProducerExecutor, 1153
  - receive, 1147, 1148

- RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT, 1154
- RECEIVE\_TIMEOUT\_NO\_WAIT, 1154
- ReceiveExecutor, 1153
- receiveSelected, 1148, 1149
- ResolveProducerExecutor, 1153
- ResolveReceiveExecutor, 1153
- send, 1149, 1150
- SendExecutor, 1153
- setDefaultDestination, 1150
- setDefaultDestinationName, 1151
- setDeliveryMode, 1151
- setDeliveryPersistent, 1151
- setExplicitQosEnabled, 1152
- setMessageIdEnabled, 1152
- setMessageTimestampEnabled, 1152
- setNoLocal, 1152
- setPriority, 1152
- setPubSubDomain, 1152
- setReceiveTimeout, 1153
- setTimeToLive, 1153
- activemq::cmsutil::CmsTemplate::ProducerExecutor, 3013
  - ~ProducerExecutor, 3013
  - action, 3014
  - destination, 3014
  - doInCms, 3014
  - getDestination, 3014
  - operator=, 3014
  - parent, 3014
  - ProducerExecutor, 3013
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3119
  - ~ReceiveExecutor, 3120
  - destination, 3121
  - doInCms, 3120
  - getDestination, 3120
  - getMessage, 3121
  - message, 3121
  - noLocal, 3121
  - operator=, 3121
  - parent, 3121
  - ReceiveExecutor, 3120
  - selector, 3121
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3221
  - ~ResolveProducerExecutor, 3221
  - getDestination, 3222
  - operator=, 3222
  - ResolveProducerExecutor, 3221
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3222
  - ~ResolveReceiveExecutor, 3222
  - getDestination, 3223
  - operator=, 3223
  - ResolveReceiveExecutor, 3222
- activemq::cmsutil::CmsTemplate::SendExecutor, 3290
  - ~SendExecutor, 3291
  - doInCms, 3291
  - operator=, 3291
  - SendExecutor, 3291
- activemq::cmsutil::DestinationResolver, 1720
  - ~DestinationResolver, 1720
  - destroy, 1721
  - init, 1721
  - resolveDestinationName, 1721
- activemq::cmsutil::DynamicDestinationResolver, 1786
  - ~DynamicDestinationResolver, 1787
  - destroy, 1787
  - DynamicDestinationResolver, 1787
  - init, 1787
  - operator=, 1787
  - resolveDestinationName, 1787
- activemq::cmsutil::MessageCreator, 2554
  - ~MessageCreator, 2554
  - createMessage, 2554
- activemq::cmsutil::PooledSession, 2904
  - ~PooledSession, 2907
  - close, 2907
  - commit, 2907
  - createBrowser, 2908
  - createBytesMessage, 2909
  - createCachedConsumer, 2909
  - createCachedProducer, 2910
  - createConsumer, 2910, 2911
  - createDurableConsumer, 2912
  - createMapMessage, 2912
  - createMessage, 2913
  - createProducer, 2913
  - createQueue, 2913
  - createStreamMessage, 2914
  - createTemporaryQueue, 2914
  - createTemporaryTopic, 2914
  - createTextMessage, 2915
  - createTopic, 2915
  - getAcknowledgeMode, 2916
  - getSession, 2916
  - isTransacted, 2916

- operator=, 2917
- PooledSession, 2907
- recover, 2917
- rollback, 2917
- unsubscribe, 2918
- activemq::cmsutil::ProducerCallback, 3012
  - ~ProducerCallback, 3012
  - doInCms, 3012
- activemq::cmsutil::ResourceLifecycleManager, 3224
  - ~ResourceLifecycleManager, 3225
  - addConnection, 3226
  - addDestination, 3226
  - addMessageConsumer, 3226
  - addMessageProducer, 3226
  - addSession, 3226
  - destroy, 3227
  - operator=, 3227
  - releaseAll, 3227
  - ResourceLifecycleManager, 3225
- activemq::cmsutil::SessionCallback, 3319
  - ~SessionCallback, 3320
  - doInCms, 3320
- activemq::cmsutil::SessionPool, 3376
  - ~SessionPool, 3377
  - getResourceLifecycleManager, 3377
  - operator=, 3377
  - returnSession, 3377
  - SessionPool, 3376
  - takeSession, 3377
- activemq::commands, 95
- activemq::commands::ActiveMQBlobMessage, 172
  - ~ActiveMQBlobMessage, 174
  - ActiveMQBlobMessage, 174
  - BINARY\_MIME\_TYPE, 177
  - clone, 174
  - cloneDataStructure, 174
  - copyDataStructure, 174
  - equals, 174
  - getDataStructureType, 175
  - getMimeType, 175
  - getName, 175
  - getRemoteBlobUrl, 175
  - ID\_ACTIVEMQBLOBMESSAGE, 177
  - isDeletedByBroker, 176
  - setDeletedByBroker, 176
  - setMimeType, 176
  - setName, 176
  - setRemoteBlobUrl, 176
  - toString, 177
- activemq::commands::ActiveMQBytesMessage, 202
  - ~ActiveMQBytesMessage, 204
  - ActiveMQBytesMessage, 204
  - clearBody, 205
  - clone, 205
  - cloneDataStructure, 205
  - copyDataStructure, 205
  - equals, 205
  - getBodyBytes, 206
  - getBodyLength, 206
  - getDataStructureType, 207
  - ID\_ACTIVEMQBYTESMESSAGE, 220
  - onSend, 207
  - readBoolean, 207
  - readByte, 207
  - readBytes, 208, 209
  - readChar, 209
  - readDouble, 210
  - readFloat, 210
  - readInt, 211
  - readLong, 211
  - readShort, 212
  - readString, 212
  - readUnsignedShort, 213
  - readUTF, 213
  - reset, 213
  - setBodyBytes, 214
  - toString, 214
  - writeBoolean, 214
  - writeByte, 215
  - writeBytes, 215, 216
  - writeChar, 216
  - writeDouble, 217
  - writeFloat, 217
  - writeInt, 217
  - writeLong, 218
  - writeShort, 218
  - writeString, 219
  - writeUnsignedShort, 219
  - writeUTF, 219
- activemq::commands::ActiveMQDestination, 293
  - ~ActiveMQDestination, 296
  - ActiveMQDestination, 296
  - advisory, 302
  - ADVISORY\_PREFIX, 302
  - cloneDataStructure, 296
  - COMPOSITE\_SEPARATOR, 303

CONNECTION\_ADVISORY\_PREFIX, `activemq::commands::ActiveMQMapMessage`, 303  
 CONSUMER\_ADVISORY\_PREFIX, 303  
 copyDataStructure, 296  
 createDestination, 296  
 createTemporaryName, 297  
 DEFAULT\_ORDERED\_TARGET, 303  
 equals, 297  
 exclusive, 303  
 getClientId, 297  
 getCMSDestination, 298  
 getDataStructureType, 298  
 getDestinationType, 298  
 getOptions, 299  
 getOrderedTarget, 299  
 getPhysicalName, 299  
 ID\_ACTIVEMQDESTINATION, 303  
 isAdvisory, 299  
 isComposite, 299  
 isConnectionAdvisory, 299  
 isConsumerAdvisory, 300  
 isExclusive, 300  
 isOrdered, 300  
 isProducerAdvisory, 300  
 isQueue, 300  
 isTemporary, 300  
 isTopic, 301  
 isWildcard, 301  
 options, 303  
 ordered, 303  
 orderedTarget, 303  
 physicalName, 303  
 PRODUCER\_ADVISORY\_PREFIX, 303  
 QUEUE\_QUALIFIED\_PREFIX, 303  
 setAdvisory, 301  
 setExclusive, 301  
 setOrdered, 301  
 setOrderedTarget, 302  
 setPhysicalName, 302  
 TEMP\_POSTFIX, 304  
 TEMP\_PREFIX, 304  
 TEMP\_QUEUE\_QUALIFIED\_PREFIX, 304  
 TEMP\_TOPIC\_QUALIFIED\_PREFIX, 304  
 TOPIC\_QUALIFIED\_PREFIX, 304  
 toString, 302  
`activemq::commands::ActiveMQDestination::DestinationFilter`, 1691  
 ANY\_CHILD, 1691  
 ANY\_DESCENDENT, 1691  
`activemq::commands::ActiveMQMapMessage`, 330  
 ~ActiveMQMapMessage, 333  
 ActiveMQMapMessage, 333  
 beforeMarshal, 333  
 checkMapsUnmarshalled, 333  
 clearBody, 333  
 clone, 333  
 cloneDataStructure, 334  
 copyDataStructure, 334  
 equals, 334  
 getBoolean, 334  
 getByte, 335  
 getBytes, 335  
 getChar, 336  
 getDataStructureType, 336  
 getDouble, 336  
 getFloat, 337  
 getInt, 337  
 getLong, 337  
 getMap, 338  
 getMapNames, 338  
 getShort, 338  
 getString, 339  
 ID\_ACTIVEMQMAPMESSAGE, 344  
 isMarshalAware, 339  
 itemExists, 339  
 setBoolean, 340  
 setByte, 340  
 setBytes, 340  
 setChar, 341  
 setDouble, 341  
 setFloat, 342  
 setInt, 342  
 setLong, 342  
 setShort, 343  
 setString, 343  
 toString, 344  
`activemq::commands::ActiveMQMessage`, 368  
 ~ActiveMQMessage, 369  
 ActiveMQMessage, 369  
 clone, 369  
 cloneDataStructure, 369  
 copyDataStructure, 370  
 getDataStructureType, 370  
 ID\_ACTIVEMQMESSAGE, 371  
 toString, 370

- activemq::commands::ActiveMQMessageTemplate, 395
  - ~ActiveMQMessageTemplate, 398
  - acknowledge, 398
  - ActiveMQMessageTemplate, 398
  - clearBody, 398
  - clearProperties, 399
  - equals, 399
  - failIfReadOnlyBody, 399
  - failIfReadOnlyProperties, 399
  - failIfWriteOnlyBody, 399
  - getBooleanProperty, 399
  - getByteProperty, 400
  - getCMSCorrelationID, 400
  - getCMSDeliveryMode, 400
  - getCMSDestination, 401
  - getCMSExpiration, 401
  - getCMSMessageID, 401
  - getCMSPriority, 402
  - getCMSRedelivered, 402
  - getCMSReplyTo, 402
  - getCMSTimestamp, 403
  - getCMSType, 403
  - getDoubleProperty, 403
  - getFloatProperty, 404
  - getIntProperty, 404
  - getLongProperty, 405
  - getPropertyNames, 405
  - getShortProperty, 406
  - getStringProperty, 406
  - onSend, 407
  - propertyExists, 407
  - setBooleanProperty, 407
  - setByteProperty, 408
  - setCMSCorrelationID, 408
  - setCMSDeliveryMode, 408
  - setCMSDestination, 408
  - setCMSExpiration, 409
  - setCMSMessageID, 409
  - setCMSPriority, 409
  - setCMSRedelivered, 410
  - setCMSReplyTo, 410
  - setCMSTimestamp, 410
  - setCMSType, 411
  - setDoubleProperty, 411
  - setFloatProperty, 411
  - setIntProperty, 412
  - setLongProperty, 412
  - setShortProperty, 413
  - setStringProperty, 413
- activemq::commands::ActiveMQObjectMessage, 414
  - ~ActiveMQObjectMessage, 414
  - ActiveMQObjectMessage, 414
  - clone, 415
  - cloneDataStructure, 415
  - copyDataStructure, 415
  - equals, 415
  - getDataStructureType, 416
  - ID\_ACTIVEMQOBJECTMESSAGE, 416
  - toString, 416
- activemq::commands::ActiveMQQueue, 453
  - ~ActiveMQQueue, 454
  - ActiveMQQueue, 454
  - clone, 454
  - cloneDataStructure, 454
  - copy, 454
  - copyDataStructure, 455
  - equals, 455
  - getCMSDestination, 455
  - getCMSProperties, 455
  - getDataStructureType, 456
  - getDestinationType, 456
  - getQueueName, 456
  - ID\_ACTIVEMQQUEUE, 457
  - toString, 456
- activemq::commands::ActiveMQStreamMessage, 506
  - ~ActiveMQStreamMessage, 509
  - ActiveMQStreamMessage, 509
  - clearBody, 509
  - clone, 509
  - cloneDataStructure, 509
  - copyDataStructure, 510
  - equals, 510
  - getDataStructureType, 510
  - ID\_ACTIVEMQSTREAMMESSAGE, 523
  - onSend, 510
  - readBoolean, 511
  - readByte, 511
  - readBytes, 512
  - readChar, 513
  - readDouble, 514
  - readFloat, 514
  - readInt, 515
  - readLong, 515
  - readShort, 516
  - readString, 516
  - readUnsignedShort, 517
  - reset, 517

- toString, 517
- writeBoolean, 518
- writeByte, 518
- writeBytes, 519
- writeChar, 519
- writeDouble, 520
- writeFloat, 520
- writeInt, 521
- writeLong, 521
- writeShort, 521
- writeString, 522
- writeUnsignedShort, 522
- activemq::commands::ActiveMQTempDestination, ~ActiveMQTempDestination, 547
  - ~ActiveMQTempDestination, 548
  - ActiveMQTempDestination, 548
  - cloneDataStructure, 548
  - close, 549
  - connection, 550
  - copyDataStructure, 549
  - equals, 549
  - getDataStructureType, 549
  - ID\_ACTIVEMQTEMPDESTINATION, 550
  - setConnection, 550
  - toString, 550
- activemq::commands::ActiveMQTempQueue, 574
  - ~ActiveMQTempQueue, 575
  - ActiveMQTempQueue, 575
  - clone, 575
  - cloneDataStructure, 575
  - copy, 575
  - copyDataStructure, 576
  - destroy, 576
  - equals, 576
  - getCMSDestination, 576
  - getCMSProperties, 577
  - getDataStructureType, 577
  - getDestinationType, 577
  - getQueueName, 577
  - ID\_ACTIVEMQTEMPQUEUE, 578
  - toString, 578
- activemq::commands::ActiveMQTempTopic, 602
  - ~ActiveMQTempTopic, 603
  - ActiveMQTempTopic, 603
  - clone, 603
  - cloneDataStructure, 604
  - copy, 604
  - copyDataStructure, 604
  - destroy, 604
  - equals, 605
  - getCMSDestination, 605
  - getCMSProperties, 605
  - getDataStructureType, 605
  - getDestinationType, 606
  - getTopicName, 606
  - ID\_ACTIVEMQTEMPTOPIC, 606
  - toString, 606
- activemq::commands::ActiveMQTextMessage, 631
  - ActiveMQTextMessage, 632
  - beforeMarshal, 632
  - clearBody, 632
  - clone, 632
  - cloneDataStructure, 633
  - copyDataStructure, 633
  - equals, 633
  - getDataStructureType, 633
  - getSize, 634
  - getText, 634
  - ID\_ACTIVEMQTEXTMESSAGE, 635
  - setText, 634
  - text, 635
  - toString, 635
- activemq::commands::ActiveMQTopic, 660
  - ~ActiveMQTopic, 661
  - ActiveMQTopic, 661
  - clone, 661
  - cloneDataStructure, 661
  - copy, 661
  - copyDataStructure, 661
  - equals, 662
  - getCMSDestination, 662
  - getCMSProperties, 662
  - getDataStructureType, 662
  - getDestinationType, 663
  - getTopicName, 663
  - ID\_ACTIVEMQTOPIC, 664
  - toString, 663
- activemq::commands::BaseCommand, 723
  - ~BaseCommand, 724
  - BaseCommand, 724
  - copyDataStructure, 724
  - equals, 725
  - getCommandId, 726
  - isBrokerInfo, 726
  - isConnectionInfo, 726

- isConsumerInfo, 726
- isKeepAliveInfo, 726
- isMessage, 727
- isMessageAck, 727
- isMessageDispatch, 727
- isMessageDispatchNotification, 727
- isProducerAck, 727
- isProducerInfo, 727
- isRemoveInfo, 727
- isRemoveSubscriptionInfo, 728
- isResponse, 728
- isResponseRequired, 728
- isShutdownInfo, 728
- isTransactionInfo, 728
- isWireFormatInfo, 728
- setCommandId, 729
- setResponseRequired, 729
- toString, 729
- activemq::commands::BaseDataStructure, 793
  - ~BaseDataStructure, 794
  - afterMarshal, 794
  - afterUnmarshal, 794
  - beforeMarshal, 794
  - beforeUnmarshal, 794
  - copyDataStructure, 795
  - equals, 795
  - getMarshaledForm, 795
  - isMarshalAware, 796
  - setMarshaledForm, 796
  - toString, 796
- activemq::commands::BooleanExpression, 816
  - ~BooleanExpression, 816
  - BooleanExpression, 816
  - cloneDataStructure, 816
  - copyDataStructure, 817
  - equals, 817
  - toString, 817
- activemq::commands::BrokerError, 823
  - ~BrokerError, 824
  - BrokerError, 824
  - cloneDataStructure, 824
  - copyDataStructure, 824
  - getCause, 825
  - getDataStructureType, 825
  - getExceptionClass, 825
  - getMessage, 825
  - getStackTraceElements, 826
  - setCause, 826
  - setExceptionClass, 826
  - setMessage, 826
  - setStackTraceElements, 827
  - visit, 827
- activemq::commands::BrokerError::StackTraceElement, 3521
  - ClassName, 3521
  - FileName, 3521
  - LineNumber, 3521
  - MethodName, 3521
- activemq::commands::BrokerId, 828
  - ~BrokerId, 830
  - BrokerId, 830
  - cloneDataStructure, 830
  - COMPARATOR, 830
  - compareTo, 830
  - copyDataStructure, 830
  - equals, 830, 831
  - getDataStructureType, 831
  - getValue, 831
  - ID\_BROKERID, 832
  - operator<, 831
  - operator=, 831
  - operator==, 831
  - setValue, 831
  - toString, 831
  - value, 832
- activemq::commands::BrokerInfo, 856
  - ~BrokerInfo, 858
  - brokerId, 861
  - BrokerInfo, 858
  - brokerName, 861
  - brokerUploadUrl, 862
  - brokerURL, 862
  - cloneDataStructure, 858
  - connectionId, 862
  - copyDataStructure, 858
  - duplexConnection, 862
  - equals, 858
  - faultTolerantConfiguration, 862
  - getBrokerId, 858, 859
  - getBrokerName, 859
  - getBrokerUploadUrl, 859
  - getBrokerURL, 859
  - getConnectionId, 859
  - getDataStructureType, 859
  - getNetworkProperties, 859
  - getPeerBrokerInfos, 859
  - ID\_BROKERINFO, 862
  - isBrokerInfo, 860

- isDuplexConnection, 860
- isFaultTolerantConfiguration, 860
- isMasterBroker, 860
- isNetworkConnection, 860
- isSlaveBroker, 860
- masterBroker, 862
- networkConnection, 862
- networkProperties, 862
- peerBrokerInfos, 862
- setBrokerId, 860
- setBrokerName, 860
- setBrokerUploadUrl, 860
- setBrokerURL, 860
- setConnectionId, 860
- setDuplexConnection, 860
- setFaultTolerantConfiguration, 860
- setMasterBroker, 860
- setNetworkConnection, 861
- setNetworkProperties, 861
- setPeerBrokerInfos, 861
- setSlaveBroker, 861
- slaveBroker, 862
- toString, 861
- visit, 861
- activemq::commands::Command, 1165
  - ~Command, 1166
  - getCommandId, 1166
  - isBrokerInfo, 1167
  - isConnectionInfo, 1167
  - isConsumerInfo, 1167
  - isKeepAliveInfo, 1167
  - isMessage, 1167
  - isMessageAck, 1167
  - isMessageDispatch, 1167
  - isMessageDispatchNotification, 1167
  - isProducerAck, 1168
  - isProducerInfo, 1168
  - isRemoveInfo, 1168
  - isRemoveSubscriptionInfo, 1168
  - isResponse, 1168
  - isResponseRequired, 1168
  - isShutdownInfo, 1168
  - isTransactionInfo, 1169
  - isWireFormatInfo, 1169
  - setCommandId, 1169
  - setResponseRequired, 1169
  - toString, 1169
  - visit, 1170
- activemq::commands::ConnectionControl, 1237
  - ~ConnectionControl, 1238
- cloneDataStructure, 1238
- close, 1241
- connectedBrokers, 1241
- ConnectionControl, 1238
- copyDataStructure, 1239
- equals, 1239
- exit, 1241
- faultTolerant, 1241
- getConnectedBrokers, 1239
- getDataStructureType, 1239
- getReconnectTo, 1239
- ID\_CONNECTIONCONTROL, 1241
- isClose, 1240
- isExit, 1240
- isFaultTolerant, 1240
- isRebalanceConnection, 1240
- isResume, 1240
- isSuspend, 1240
- rebalanceConnection, 1241
- reconnectTo, 1241
- resume, 1241
- setClose, 1240
- setConnectedBrokers, 1240
- setExit, 1240
- setFaultTolerant, 1240
- setRebalanceConnection, 1240
- setReconnectTo, 1240
- setResume, 1240
- setSuspend, 1240
- suspend, 1242
- toString, 1240
- visit, 1241
- activemq::commands::ConnectionError, 1266
  - ~ConnectionError, 1267
  - cloneDataStructure, 1267
  - ConnectionError, 1267
  - connectionId, 1269
  - copyDataStructure, 1267
  - equals, 1268
  - exception, 1269
  - getConnectionId, 1268
  - getDataStructureType, 1268
  - getException, 1268
  - ID\_CONNECTIONERROR, 1269
  - setConnectionId, 1268
  - setException, 1268
  - toString, 1269
  - visit, 1269
- activemq::commands::ConnectionId, 1297
  - ~ConnectionId, 1298



- cloneDataStructure, 1298
- COMPARATOR, 1298
- compareTo, 1299
- ConnectionId, 1298
- copyDataStructure, 1299
- equals, 1299
- getDataStructureType, 1299
- getValue, 1299, 1300
- ID\_CONNECTIONID, 1300
- operator<, 1300
- operator=, 1300
- operator==, 1300
- setValue, 1300
- toString, 1300
- value, 1300
- activemq::commands::ConnectionInfo, 1324
  - ~ConnectionInfo, 1326
  - brokerMasterConnector, 1330
  - brokerPath, 1330
  - clientId, 1330
  - clientMaster, 1330
  - cloneDataStructure, 1326
  - connectionId, 1330
  - ConnectionInfo, 1326
  - copyDataStructure, 1326
  - createRemoveCommand, 1327
  - equals, 1327
  - faultTolerant, 1330
  - getBrokerPath, 1327
  - getClientId, 1327
  - getConnectionId, 1327
  - getDataStructureType, 1327
  - getPassword, 1328
  - getUserName, 1328
  - ID\_CONNECTIONINFO, 1330
  - isBrokerMasterConnector, 1328
  - isClientMaster, 1328
  - isConnectionInfo, 1328
  - isFaultTolerant, 1328
  - isManageable, 1328
  - manageable, 1330
  - password, 1330
  - setBrokerMasterConnector, 1328
  - setBrokerPath, 1328
  - setClientId, 1329
  - setClientMaster, 1329
  - setConnectionId, 1329
  - setFaultTolerant, 1329
  - setManageable, 1329
  - setPassword, 1329
  - setUserName, 1329
  - toString, 1329
  - userName, 1330
  - visit, 1329
- activemq::commands::ConsumerControl, 1369
  - ~ConsumerControl, 1370
  - cloneDataStructure, 1370
  - close, 1373
  - ConsumerControl, 1370
  - consumerId, 1373
  - copyDataStructure, 1370
  - destination, 1373
  - equals, 1371
  - flush, 1373
  - getConsumerId, 1371
  - getDataStructureType, 1371
  - getDestination, 1371
  - getPrefetch, 1372
  - ID\_CONSUMERCONTROL, 1373
  - isClose, 1372
  - isFlush, 1372
  - isStart, 1372
  - isStop, 1372
  - prefetch, 1373
  - setClose, 1372
  - setConsumerId, 1372
  - setDestination, 1372
  - setFlush, 1372
  - setPrefetch, 1372
  - setStart, 1372
  - setStop, 1372
  - start, 1373
  - stop, 1373
  - toString, 1372
  - visit, 1372
- activemq::commands::ConsumerId, 1398
  - ~ConsumerId, 1399
  - cloneDataStructure, 1399
  - COMPARATOR, 1399
  - compareTo, 1399
  - connectionId, 1401
  - ConsumerId, 1399
  - copyDataStructure, 1399
  - equals, 1400
  - getConnectionId, 1400
  - getDataStructureType, 1400
  - getParentId, 1400
  - getSessionId, 1400
  - getValue, 1401
  - ID\_CONSUMERID, 1401

- operator<, 1401
- operator=, 1401
- operator==, 1401
- sessionId, 1401
- setConnectionId, 1401
- setSessionId, 1401
- setValue, 1401
- toString, 1401
- value, 1402
- activemq::commands::ConsumerInfo, 1426
  - ~ConsumerInfo, 1428
  - additionalPredicate, 1433
  - brokerPath, 1433
  - browser, 1433
  - cloneDataStructure, 1428
  - consumerId, 1433
  - ConsumerInfo, 1428
  - copyDataStructure, 1428
  - createRemoveCommand, 1429
  - destination, 1433
  - dispatchAsync, 1433
  - equals, 1429
  - exclusive, 1433
  - getAdditionalPredicate, 1429
  - getBrokerPath, 1429
  - getConsumerId, 1429
  - getDataStructureType, 1430
  - getDestination, 1430
  - getMaximumPendingMessageLimit, 1430
  - getNetworkConsumerPath, 1430
  - getPrefetchSize, 1430
  - getPriority, 1430
  - getSelector, 1430
  - getSubscriptionName, 1430
  - ID\_CONSUMERINFO, 1433
  - isBrowser, 1431
  - isConsumerInfo, 1431
  - isDispatchAsync, 1431
  - isExclusive, 1431
  - isNetworkSubscription, 1431
  - isNoLocal, 1431
  - isNoRangeAcks, 1431
  - isOptimizedAcknowledge, 1431
  - isRetroactive, 1431
  - maximumPendingMessageLimit, 1433
  - networkConsumerPath, 1433
  - networkSubscription, 1434
  - noLocal, 1434
  - noRangeAcks, 1434
  - optimizedAcknowledge, 1434
  - prefetchSize, 1434
  - priority, 1434
  - retroactive, 1434
  - selector, 1434
  - setAdditionalPredicate, 1431
  - setBrokerPath, 1431
  - setBrowser, 1431
  - setConsumerId, 1431
  - setDestination, 1431
  - setDispatchAsync, 1432
  - setExclusive, 1432
  - setMaximumPendingMessageLimit, 1432
  - setNetworkConsumerPath, 1432
  - setNetworkSubscription, 1432
  - setNoLocal, 1432
  - setNoRangeAcks, 1432
  - setOptimizedAcknowledge, 1432
  - setPrefetchSize, 1432
  - setPriority, 1432
  - setRetroactive, 1432
  - setSelector, 1432
  - setSubscriptionName, 1432
  - subscriptionName, 1434
  - toString, 1432
  - visit, 1433
- activemq::commands::ControlCommand, 1459
  - ~ControlCommand, 1460
  - cloneDataStructure, 1460
  - command, 1462
  - ControlCommand, 1460
  - copyDataStructure, 1460
  - equals, 1461
  - getCommand, 1461
  - getDataStructureType, 1461
  - ID\_CONTROLCOMMAND, 1462
  - setCommand, 1461
  - toString, 1461
  - visit, 1462
- activemq::commands::DataArrayResponse, 1493
  - ~DataArrayResponse, 1494
  - cloneDataStructure, 1494
  - copyDataStructure, 1494
  - data, 1496
  - DataArrayResponse, 1494
  - equals, 1495
  - getData, 1495
  - getDataStructureType, 1495
  - ID\_DATAARRAYRESPONSE, 1496
  - setData, 1495

- toString, 1495
- activemq::commands::DataResponse, 1550
  - ~DataResponse, 1551
  - cloneDataStructure, 1551
  - copyDataStructure, 1551
  - data, 1552
  - DataResponse, 1551
  - equals, 1551
  - getData, 1552
  - getDataStructureType, 1552
  - ID\_DATARESPONSE, 1552
  - setData, 1552
  - toString, 1552
- activemq::commands::DataStructure, 1628
  - ~DataStructure, 1628
  - cloneDataStructure, 1628
  - copyDataStructure, 1629
  - equals, 1630
  - getDataStructureType, 1631
  - toString, 1632
- activemq::commands::DestinationInfo, 1691
  - ~DestinationInfo, 1693
  - brokerPath, 1695
  - cloneDataStructure, 1693
  - connectionId, 1695
  - copyDataStructure, 1693
  - destination, 1695
  - DestinationInfo, 1693
  - equals, 1693
  - getBrokerPath, 1694
  - getConnectionId, 1694
  - getDataStructureType, 1694
  - getDestination, 1694
  - getOperationType, 1694
  - getTimeout, 1694
  - ID\_DESTINATIONINFO, 1696
  - operationType, 1696
  - setBrokerPath, 1694
  - setConnectionId, 1694
  - setDestination, 1695
  - setOperationType, 1695
  - setTimeout, 1695
  - timeout, 1696
  - toString, 1695
  - visit, 1695
- activemq::commands::DiscoveryEvent, 1722
  - ~DiscoveryEvent, 1723
  - brokerName, 1724
  - cloneDataStructure, 1723
  - copyDataStructure, 1723
  - DiscoveryEvent, 1723
  - equals, 1723
  - getBrokerName, 1724
  - getDataStructureType, 1724
  - getServiceName, 1724
  - ID\_DISCOVERYEVENT, 1724
  - serviceName, 1725
  - setBrokerName, 1724
  - setServiceName, 1724
  - toString, 1724
- activemq::commands::ExceptionResponse, 1802
  - ~ExceptionResponse, 1802
  - cloneDataStructure, 1803
  - copyDataStructure, 1803
  - equals, 1803
  - exception, 1804
  - ExceptionResponse, 1802
  - getDataStructureType, 1803
  - getException, 1804
  - ID\_EXCEPTIONRESPONSE, 1804
  - setException, 1804
  - toString, 1804
- activemq::commands::FlushCommand, 1900
  - ~FlushCommand, 1901
  - cloneDataStructure, 1901
  - copyDataStructure, 1901
  - equals, 1902
  - FlushCommand, 1901
  - getDataStructureType, 1902
  - ID\_FLUSHCOMMAND, 1903
  - toString, 1902
  - visit, 1902
- activemq::commands::IntegerResponse, 2054
  - ~IntegerResponse, 2055
  - cloneDataStructure, 2055
  - copyDataStructure, 2055
  - equals, 2055
  - getDataStructureType, 2056
  - getResult, 2056
  - ID\_INTEGERRESPONSE, 2056
  - IntegerResponse, 2055
  - result, 2056
  - setResult, 2056
  - toString, 2056
- activemq::commands::JournalQueueAck, 2116
  - ~JournalQueueAck, 2117
  - cloneDataStructure, 2117
  - copyDataStructure, 2117
  - destination, 2119

- equals, 2117
  - getDataStructureType, 2118
  - getDestination, 2118
  - getMessageAck, 2118
  - ID\_JOURNALQUEUEACK, 2119
  - JournalQueueAck, 2117
  - messageAck, 2119
  - setDestination, 2118
  - setMessageAck, 2118
  - toString, 2118
- activemq::commands::JournalTopicAck, 2143
  - ~JournalTopicAck, 2144
  - clientId, 2147
  - cloneDataStructure, 2144
  - copyDataStructure, 2145
  - destination, 2147
  - equals, 2145
  - getClientId, 2145
  - getDataStructureType, 2145
  - getDestination, 2146
  - getMessageId, 2146
  - getMessageSequenceId, 2146
  - getSubscriptionName, 2146
  - getTransactionId, 2146
  - ID\_JOURNALTOPICACK, 2147
  - JournalTopicAck, 2144
  - messageId, 2147
  - messageSequenceId, 2147
  - setClientId, 2146
  - setDestination, 2146
  - setMessageId, 2146
  - setMessageSequenceId, 2146
  - setSubscriptionName, 2146
  - setTransactionId, 2147
  - subscriptionName, 2147
  - toString, 2147
  - transactionId, 2147
- activemq::commands::JournalTrace, 2171
  - ~JournalTrace, 2172
  - cloneDataStructure, 2172
  - copyDataStructure, 2173
  - equals, 2173
  - getDataStructureType, 2173
  - getMessage, 2173, 2174
  - ID\_JOURNALTRACE, 2174
  - JournalTrace, 2172
  - message, 2174
  - setMessage, 2174
  - toString, 2174
- activemq::commands::JournalTransaction, 2198
  - ~JournalTransaction, 2199
  - cloneDataStructure, 2199
  - copyDataStructure, 2199
  - equals, 2200
  - getDataStructureType, 2200
  - getTransactionId, 2200
  - getType, 2200
  - getWasPrepared, 2200
  - ID\_JOURNALTRANSACTION, 2201
  - JournalTransaction, 2199
  - setTransactionId, 2201
  - setType, 2201
  - setWasPrepared, 2201
  - toString, 2201
  - transactionId, 2201
  - type, 2201
  - wasPrepared, 2201
- activemq::commands::KeepAliveInfo, 2225
  - ~KeepAliveInfo, 2226
  - cloneDataStructure, 2226
  - copyDataStructure, 2226
  - equals, 2227
  - getDataStructureType, 2227
  - ID\_KEEPAALIVEINFO, 2228
  - isKeepAliveInfo, 2227
  - KeepAliveInfo, 2226
  - toString, 2227
  - visit, 2228
- activemq::commands::LastPartialCommand, 2260
  - ~LastPartialCommand, 2261
  - cloneDataStructure, 2261
  - copyDataStructure, 2261
  - equals, 2261
  - getDataStructureType, 2262
  - ID\_LASTPARTIALCOMMAND, 2262
  - LastPartialCommand, 2261
  - toString, 2262
- activemq::commands::LocalTransactionId, 2306
  - ~LocalTransactionId, 2308
  - cloneDataStructure, 2308
  - COMPARATOR, 2308
  - compareTo, 2308
  - connectionId, 2310
  - copyDataStructure, 2308
  - equals, 2308, 2309
  - getConnectionId, 2309
  - getDataStructureType, 2309

- getValue, 2309
- ID\_LOCALTRANSACTIONID, 2310
- LocalTransactionId, 2308
- operator<, 2309
- operator=, 2309
- operator==, 2309
- setConnectionId, 2309
- setValue, 2310
- toString, 2310
- value, 2310
- activemq::commands::Message, 2475
  - ~Message, 2480
  - afterUnmarshal, 2480
  - arrival, 2490
  - beforeMarshal, 2480
  - brokerInTime, 2491
  - brokerOutTime, 2491
  - brokerPath, 2491
  - cloneDataStructure, 2480
  - cluster, 2491
  - compressed, 2491
  - connection, 2491
  - content, 2491
  - copyDataStructure, 2481
  - correlationId, 2491
  - dataStructure, 2491
  - DEFAULT\_MESSAGE\_SIZE, 2491
  - destination, 2491
  - droppable, 2491
  - equals, 2481
  - expiration, 2491
  - getAckHandler, 2481
  - getArrival, 2482
  - getBrokerInTime, 2482
  - getBrokerOutTime, 2482
  - getBrokerPath, 2482
  - getCluster, 2482
  - getConnection, 2482
  - getContent, 2482
  - getCorrelationId, 2482, 2483
  - getDataStructure, 2483
  - getDataStructureType, 2483
  - getDestination, 2483
  - getExpiration, 2483
  - getGroupId, 2483
  - getGroupSequence, 2483
  - getMarshaledProperties, 2483
  - getMessageId, 2484
  - getMessageProperties, 2484
  - getOriginalDestination, 2484
  - getOriginalTransactionId, 2484
  - getPriority, 2484
  - getProducerId, 2484
  - getRedeliveryCounter, 2484
  - getReplyTo, 2485
  - getSize, 2485
  - getTargetConsumerId, 2485
  - getTimestamp, 2485
  - getTransactionId, 2485
  - getType, 2485
  - getUserID, 2485
  - groupId, 2491
  - groupSequence, 2491
  - ID\_MESSAGE, 2491
  - isCompressed, 2485
  - isDroppable, 2486
  - isExpired, 2486
  - isMarshalAware, 2486
  - isMessage, 2486
  - isPersistent, 2486
  - isReadOnlyBody, 2486
  - isReadOnlyProperties, 2486
  - isRecievedByDFBridge, 2487
  - marshalledProperties, 2491
  - Message, 2480
  - messageId, 2492
  - onSend, 2487
  - originalDestination, 2492
  - originalTransactionId, 2492
  - persistent, 2492
  - priority, 2492
  - producerId, 2492
  - recievedByDFBridge, 2492
  - redeliveryCounter, 2492
  - replyTo, 2492
  - setAckHandler, 2487
  - setArrival, 2487
  - setBrokerInTime, 2487
  - setBrokerOutTime, 2487
  - setBrokerPath, 2487
  - setCluster, 2488
  - setCompressed, 2488
  - setConnection, 2488
  - setContent, 2488
  - setCorrelationId, 2488
  - setDataStructure, 2488
  - setDestination, 2488
  - setDroppable, 2488
  - setExpiration, 2488
  - setGroupId, 2488

- setGroupSequence, 2488
- setMarshaledProperties, 2488
- setMessageId, 2488
- setOriginalDestination, 2489
- setOriginalTransactionId, 2489
- setPersistent, 2489
- setPriority, 2489
- setProducerId, 2489
- setReadOnlyBody, 2489
- setReadOnlyProperties, 2489
- setReceivedByDFBridge, 2489
- setRedeliveryCounter, 2489
- setReplyTo, 2489
- setTargetConsumerId, 2489
- setTimestamp, 2490
- setTransactionId, 2490
- setType, 2490
- setUserId, 2490
- targetConsumerId, 2492
- timestamp, 2490
- toString, 2490
- transactionId, 2492
- type, 2492
- userId, 2492
- visit, 2490
- activemq::commands::MessageAck, 2521
  - ~MessageAck, 2522
  - ackType, 2525
  - cloneDataStructure, 2522
  - consumerId, 2525
  - copyDataStructure, 2522
  - destination, 2525
  - equals, 2523
  - firstMessageId, 2525
  - getAckType, 2523
  - getConsumerId, 2523
  - getDataStructureType, 2523
  - getDestination, 2523
  - getFirstMessageId, 2523, 2524
  - getLastMessageId, 2524
  - getMessageCount, 2524
  - getTransactionId, 2524
  - ID\_MESSAGEACK, 2526
  - isMessageAck, 2524
  - lastMessageId, 2526
  - MessageAck, 2522
  - messageCount, 2526
  - setAckType, 2524
  - setConsumerId, 2524
  - setDestination, 2524
  - setFirstMessageId, 2524
  - setLastMessageId, 2524
  - setMessageCount, 2525
  - setTransactionId, 2525
  - toString, 2525
  - transactionId, 2526
  - visit, 2525
- activemq::commands::MessageDispatch, 2555
  - ~MessageDispatch, 2556
  - cloneDataStructure, 2556
  - consumerId, 2559
  - copyDataStructure, 2556
  - destination, 2559
  - equals, 2557
  - getConsumerId, 2557
  - getDataStructureType, 2557
  - getDestination, 2557
  - getMessage, 2557
  - getRedeliveryCounter, 2558
  - ID\_MESSAGEDISPATCH, 2559
  - isMessageDispatch, 2558
  - message, 2559
  - MessageDispatch, 2556
  - redeliveryCounter, 2559
  - setConsumerId, 2558
  - setDestination, 2558
  - setMessage, 2558
  - setRedeliveryCounter, 2558
  - toString, 2558
  - visit, 2558
- activemq::commands::MessageDispatchNotification, 2590
  - ~MessageDispatchNotification, 2592
  - cloneDataStructure, 2592
  - consumerId, 2594
  - copyDataStructure, 2592
  - deliverySequenceId, 2594
  - destination, 2594
  - equals, 2592
  - getConsumerId, 2592, 2593
  - getDataStructureType, 2593
  - getDeliverySequenceId, 2593
  - getDestination, 2593
  - getMessageId, 2593
  - ID\_MESSAGEDISPATCHNOTIFICATION, 2595
  - isMessageDispatchNotification, 2593
  - MessageDispatchNotification, 2592
  - messageId, 2595
  - setConsumerId, 2593

- setDeliverySequenceId, 2594
- setDestination, 2594
- setMessageId, 2594
- toString, 2594
- visit, 2594
- activemq::commands::MessageId, 2623
  - ~MessageId, 2625
  - brokerSequenceId, 2627
  - cloneDataStructure, 2625
  - COMPARATOR, 2625
  - compareTo, 2625
  - copyDataStructure, 2626
  - equals, 2626
  - getBrokerSequenceId, 2626
  - getDataStructureType, 2626
  - getProducerId, 2626, 2627
  - getProducerSequenceId, 2627
  - ID\_MESSAGEID, 2627
  - MessageId, 2625
  - operator<, 2627
  - operator=, 2627
  - operator==, 2627
  - producerId, 2628
  - producerSequenceId, 2628
  - setBrokerSequenceId, 2627
  - setProducerId, 2627
  - setProducerSequenceId, 2627
  - setTextView, 2627
  - setValue, 2627
  - toString, 2627
- activemq::commands::MessagePull, 2695
  - ~MessagePull, 2696
  - cloneDataStructure, 2696
  - consumerId, 2699
  - copyDataStructure, 2697
  - correlationId, 2699
  - destination, 2699
  - equals, 2697
  - getConsumerId, 2697
  - getCorrelationId, 2697
  - getDataStructureType, 2698
  - getDestination, 2698
  - getMessageId, 2698
  - getTimeout, 2698
  - ID\_MESSAGEPULL, 2699
  - messageId, 2699
  - MessagePull, 2696
  - setConsumerId, 2698
  - setCorrelationId, 2698
  - setDestination, 2698
- setMessageId, 2698
- setTimeout, 2698
- timeout, 2699
- toString, 2698
- visit, 2699
- activemq::commands::NetworkBridgeFilter, 2746
  - ~NetworkBridgeFilter, 2747
  - cloneDataStructure, 2747
  - copyDataStructure, 2747
  - equals, 2747
  - getDataStructureType, 2748
  - getNetworkBrokerId, 2748
  - getNetworkTTL, 2748
  - ID\_NETWORKBRIDGEFILTER, 2749
  - NetworkBridgeFilter, 2747
  - networkBrokerId, 2749
  - networkTTL, 2749
  - setNetworkBrokerId, 2748
  - setNetworkTTL, 2748
  - toString, 2748
- activemq::commands::PartialCommand, 2866
  - ~PartialCommand, 2867
  - cloneDataStructure, 2868
  - commandId, 2869
  - copyDataStructure, 2868
  - data, 2869
  - equals, 2868
  - getCommandId, 2868
  - getData, 2868, 2869
  - getDataStructureType, 2869
  - ID\_PARTIALCOMMAND, 2869
  - PartialCommand, 2867
  - setCommandId, 2869
  - setData, 2869
  - toString, 2869
- activemq::commands::ProducerAck, 2984
  - ~ProducerAck, 2985
  - cloneDataStructure, 2985
  - copyDataStructure, 2985
  - equals, 2986
  - getDataStructureType, 2986
  - getProducerId, 2986
  - getSize, 2986
  - ID\_PRODUCERACK, 2987
  - isProducerAck, 2986
  - ProducerAck, 2985
  - producerId, 2987
  - setProducerId, 2986
  - setSize, 2987

- size, 2987
- toString, 2987
- visit, 2987
- activemq::commands::ProducerId, 3014
  - ~ProducerId, 3016
  - cloneDataStructure, 3016
  - COMPARATOR, 3016
  - compareTo, 3016
  - connectionId, 3018
  - copyDataStructure, 3016
  - equals, 3017
  - getConnectionId, 3017
  - getDataStructureType, 3017
  - getParentId, 3017
  - getSessionId, 3017
  - getValue, 3017
  - ID\_PRODUCERID, 3018
  - operator<, 3017
  - operator=, 3018
  - operator==, 3018
  - ProducerId, 3016
  - sessionId, 3018
  - setConnectionId, 3018
  - setProducerSessionKey, 3018
  - setSessionId, 3018
  - setValue, 3018
  - toString, 3018
  - value, 3018
- activemq::commands::ProducerInfo, 3043
  - ~ProducerInfo, 3044
  - brokerPath, 3047
  - cloneDataStructure, 3044
  - copyDataStructure, 3044
  - createRemoveCommand, 3045
  - destination, 3047
  - dispatchAsync, 3047
  - equals, 3045
  - getBrokerPath, 3045
  - getDataStructureType, 3045
  - getDestination, 3045
  - getProducerId, 3045
  - getWindowSize, 3046
  - ID\_PRODUCERINFO, 3047
  - isDispatchAsync, 3046
  - isProducerInfo, 3046
  - producerId, 3047
  - ProducerInfo, 3044
  - setBrokerPath, 3046
  - setDestination, 3046
  - setDispatchAsync, 3046
  - setProducerId, 3046
  - setWindowSize, 3046
  - toString, 3046
  - visit, 3046
  - windowSize, 3047
- activemq::commands::RemoveInfo, 3137
  - ~RemoveInfo, 3138
  - cloneDataStructure, 3139
  - copyDataStructure, 3139
  - equals, 3139
  - getDataStructureType, 3139
  - getLastDeliveredSequenceId, 3140
  - getObjectId, 3140
  - ID\_REMOVEINFO, 3141
  - isRemoveInfo, 3140
  - lastDeliveredSequenceId, 3141
  - objectId, 3141
  - RemoveInfo, 3138
  - setLastDeliveredSequenceId, 3140
  - setObjectId, 3140
  - toString, 3140
  - visit, 3140
- activemq::commands::RemoveSubscriptionInfo, 3165
  - ~RemoveSubscriptionInfo, 3166
  - clientId, 3169
  - cloneDataStructure, 3166
  - connectionId, 3169
  - copyDataStructure, 3167
  - equals, 3167
  - getClientId, 3167
  - getConnectionId, 3167
  - getDataStructureType, 3167
  - getSubscriptionName, 3168
  - ID\_REMOVESUBSCRIPTIONINFO, 3169
  - isRemoveSubscriptionInfo, 3168
  - RemoveSubscriptionInfo, 3166
  - setClientId, 3168
  - setConnectionId, 3168
  - setSubscriptionName, 3168
  - subscriptionName, 3169
  - toString, 3168
  - visit, 3168
- activemq::commands::ReplayCommand, 3194
  - ~ReplayCommand, 3195
  - cloneDataStructure, 3195
  - copyDataStructure, 3195
  - equals, 3195
  - firstNakNumber, 3197
  - getDataStructureType, 3195



- getFirstNakNumber, 3196
- getLastNakNumber, 3196
- ID\_REPLAYCOMMAND, 3197
- lastNakNumber, 3197
- ReplayCommand, 3195
- setFirstNakNumber, 3196
- setLastNakNumber, 3196
- toString, 3196
- visit, 3196
- activemq::commands::Response, 3227
  - ~Response, 3228
  - cloneDataStructure, 3228
  - copyDataStructure, 3229
  - correlationId, 3231
  - equals, 3229
  - getCorrelationId, 3229
  - getDataStructureType, 3229
  - ID\_RESPONSE, 3231
  - isResponse, 3230
  - Response, 3228
  - setCorrelationId, 3230
  - toString, 3230
  - visit, 3230
- activemq::commands::SessionId, 3320
  - ~SessionId, 3322
  - cloneDataStructure, 3322
  - COMPARATOR, 3321
  - compareTo, 3322
  - connectionId, 3324
  - copyDataStructure, 3322
  - equals, 3322, 3323
  - getConnectionId, 3323
  - getDataStructureType, 3323
  - getParentId, 3323
  - getValue, 3323
  - ID\_SESSIONID, 3324
  - operator<, 3323
  - operator=, 3323
  - operator==, 3323
  - SessionId, 3321, 3322
  - setConnectionId, 3323
  - setValue, 3323
  - toString, 3324
  - value, 3324
- activemq::commands::SessionInfo, 3348
  - ~SessionInfo, 3349
  - cloneDataStructure, 3349
  - copyDataStructure, 3349
  - createRemoveCommand, 3350
  - equals, 3350
  - getAckMode, 3350
  - getDataStructureType, 3350
  - getSessionId, 3350
  - ID\_SESSIONINFO, 3351
  - sessionId, 3351
  - SessionInfo, 3349
  - setAckMode, 3351
  - setSessionId, 3351
  - toString, 3351
  - visit, 3351
- activemq::commands::ShutdownInfo, 3413
  - ~ShutdownInfo, 3414
  - cloneDataStructure, 3414
  - copyDataStructure, 3414
  - equals, 3414
  - getDataStructureType, 3414
  - ID\_SHUTDOWNINFO, 3416
  - isShutdownInfo, 3415
  - ShutdownInfo, 3414
  - toString, 3415
  - visit, 3415
- activemq::commands::SubscriptionInfo, 3616
  - ~SubscriptionInfo, 3617
  - clientId, 3620
  - cloneDataStructure, 3617
  - copyDataStructure, 3617
  - destination, 3620
  - equals, 3618
  - getClientId, 3618
  - getDataStructureType, 3618
  - getDestination, 3618
  - getSelector, 3618, 3619
  - getSubscriptionName, 3619
  - getSubscribedDestination, 3619
  - ID\_SUBSCRIPTIONINFO, 3620
  - selector, 3620
  - setClientId, 3619
  - setDestination, 3619
  - setSelector, 3619
  - setSubscriptionName, 3619
  - setSubscribedDestination, 3619
  - subscriptionName, 3620
  - subscribedDestination, 3620
  - SubscriptionInfo, 3617
  - toString, 3619
- activemq::commands::TransactionId, 3759
  - ~TransactionId, 3760
  - cloneDataStructure, 3761
  - COMPARATOR, 3760
  - compareTo, 3761

- copyDataStructure, 3761
- equals, 3761
- getDataStructureType, 3762
- ID\_TRANSACTIONID, 3762
- operator<, 3762
- operator=, 3762
- operator==, 3762
- toString, 3762
- TransactionId, 3760
- activemq::commands::TransactionInfo, 3785
  - ~TransactionInfo, 3786
  - cloneDataStructure, 3786
  - connectionId, 3789
  - copyDataStructure, 3786
  - equals, 3787
  - getConnectionId, 3787
  - getDataStructureType, 3787
  - getTransactionId, 3787
  - getType, 3788
  - ID\_TRANSACTIONINFO, 3789
  - isTransactionInfo, 3788
  - setConnectionId, 3788
  - setTransactionId, 3788
  - setType, 3788
  - toString, 3788
  - transactionId, 3789
  - TransactionInfo, 3786
  - type, 3789
  - visit, 3788
- activemq::commands::WireFormatInfo, 3912
  - ~WireFormatInfo, 3914
  - afterUnmarshal, 3914
  - beforeMarshal, 3915
  - cloneDataStructure, 3915
  - copyDataStructure, 3915
  - equals, 3915
  - getCacheSize, 3916
  - getDataStructureType, 3916
  - getMagic, 3916
  - getMarshaledProperties, 3916
  - getMaxInactivityDuration, 3916
  - getMaxInactivityDurationInitialDelay, 3916
  - getProperties, 3917
  - getVersion, 3917
  - ID\_WIREFORMATINFO, 3922
  - isCacheEnabled, 3917
  - isMarshalAware, 3918
  - isSizePrefixDisabled, 3918
  - isStackTraceEnabled, 3918
  - isTcpNoDelayEnabled, 3918
  - isTightEncodingEnabled, 3918
  - isValid, 3919
  - isWireFormatInfo, 3919
  - setCacheEnabled, 3919
  - setCacheSize, 3919
  - setMagic, 3919
  - setMarshaledProperties, 3920
  - setMaxInactivityDuration, 3920
  - setMaxInactivityDurationInitialDelay, 3920
  - setProperties, 3920
  - setSizePrefixDisabled, 3921
  - setStackTraceEnabled, 3921
  - setTcpNoDelayEnabled, 3921
  - setTightEncodingEnabled, 3921
  - setVersion, 3921
  - toString, 3922
  - visit, 3922
  - WireFormatInfo, 3914
- activemq::commands::XATransactionId, 3960
  - ~XATransactionId, 3962
  - branchQualifier, 3964
  - cloneDataStructure, 3962
  - COMPARATOR, 3961
  - compareTo, 3962
  - copyDataStructure, 3962
  - equals, 3962
  - formatId, 3964
  - getBranchQualifier, 3963
  - getDataStructureType, 3963
  - getFormatId, 3963
  - getGlobalTransactionId, 3963
  - globalTransactionId, 3964
  - ID\_XATRANSACTIONID, 3964
  - operator<, 3963
  - operator=, 3963
  - operator==, 3963
  - setBranchQualifier, 3963
  - setFormatId, 3963
  - setGlobalTransactionId, 3963
  - toString, 3964
  - XATransactionId, 3961
- activemq::core, 96
  - activemq::core::ActiveMQAckHandler, 171
    - ~ActiveMQAckHandler, 172
    - acknowledgeMessage, 172
  - activemq::core::ActiveMQConnection, 244
    - ~ActiveMQConnection, 249
    - ActiveMQConnection, 249
    - addDispatcher, 249
    - addProducer, 249

- addTransportListener, 250
- close, 250
- createSession, 250, 251
- destroyDestination, 251
- fire, 252
- getBrokerURL, 252
- getClientID, 252
- getCloseTimeout, 253
- getConnectionId, 253
- getConnectionInfo, 253
- getExceptionListener, 253
- getMetaData, 253
- getNextLocalTransactionId, 254
- getNextSessionId, 254
- getNextTempDestinationId, 254
- getPassword, 254
- getPrefetchPolicy, 255
- getProducerWindowSize, 255
- getRedeliveryPolicy, 255
- getSendTimeout, 255
- getTransport, 255
- getUsername, 256
- isAlwaysSyncSend, 256
- isClosed, 256
- isDispatchAsync, 256
- isStarted, 256
- isTransportFailed, 256
- isUseAsyncSend, 257
- isUseCompression, 257
- onCommand, 257
- oneway, 257
- onException, 258
- removeDispatcher, 258
- removeProducer, 258
- removeSession, 258
- removeTransportListener, 258
- sendPullRequest, 259
- setAlwaysSyncSend, 259
- setBrokerURL, 259
- setClientID, 259
- setCloseTimeout, 260
- setDefaultClientId, 260
- setDispatchAsync, 260
- setExceptionListener, 261
- setPassword, 261
- setPrefetchPolicy, 261
- setProducerWindowSize, 261
- setRedeliveryPolicy, 262
- setSendTimeout, 262
- setTransportInterruptProcessingComplete, 262
- setUseAsyncSend, 262
- setUseCompression, 263
- setUsername, 263
- start, 263
- stop, 263
- syncRequest, 263
- transportInterrupted, 264
- transportResumed, 264
- activemq::core::ActiveMQConnectionFactory, 264
  - ~ActiveMQConnectionFactory, 267
  - ActiveMQConnectionFactory, 266
  - createConnection, 267, 268
  - DEFAULT\_URI, 275
  - getBrokerURL, 269
  - getClientId, 269
  - getCloseTimeout, 269
  - getExceptionListener, 269
  - getPassword, 269
  - getPrefetchPolicy, 270
  - getProducerWindowSize, 270
  - getRedeliveryPolicy, 270
  - getSendTimeout, 270
  - getUsername, 271
  - isAlwaysSyncSend, 271
  - isDispatchAsync, 271
  - isUseAsyncSend, 271
  - isUseCompression, 271
  - setAlwaysSyncSend, 271
  - setBrokerURL, 272
  - setClientId, 272
  - setCloseTimeout, 272
  - setDispatchAsync, 272
  - setExceptionListener, 272
  - setPassword, 273
  - setPrefetchPolicy, 273
  - setProducerWindowSize, 273
  - setRedeliveryPolicy, 273
  - setSendTimeout, 274
  - setUseAsyncSend, 274
  - setUseCompression, 274
  - setUsername, 274
- activemq::core::ActiveMQConnectionMetaData, 275
  - ~ActiveMQConnectionMetaData, 276
  - ActiveMQConnectionMetaData, 276
  - getCMSMajorVersion, 276
  - getCMSMinorVersion, 276

- getCMSProviderName, 276
- getCMSVersion, 277
- getCMSXPropertyNames, 277
- getProviderMajorVersion, 277
- getProviderMinorVersion, 278
- getProviderVersion, 278
- activemq::core::ActiveMQConstants, 279
  - ACK\_TYPE\_CONSUMED, 280
  - ACK\_TYPE\_DELIVERED, 280
  - ACK\_TYPE\_INDIVIDUAL, 280
  - ACK\_TYPE\_POISON, 280
  - ACK\_TYPE\_REDELIVERED, 280
  - AckType, 280
  - CONNECTION\_ALWAYS\_SYNC\_SEND, 281
  - CONNECTION\_CLOSE\_TIMEOUT, 281
  - CONNECTION\_DISPATCH\_ASYNC, 281
  - CONNECTION\_PRODUCER\_WINDOW\_SIZE, 281
  - CONNECTION\_SEND\_TIMEOUT, 281
  - CONNECTION\_USE\_ASYNC\_SEND, 281
  - CONNECTION\_USE\_COMPRESSION, 281
  - CONSUMER\_DISPATCH\_ASYNC, 280
  - CONSUMER\_EXCLUSIVE, 281
  - CONSUMER\_NOLOCAL, 280
  - CONSUMER\_PREFETCH\_SIZE, 280
  - CONSUMER\_PRIORITY, 281
  - CONSUMER\_RETROACTIVE, 280
  - CONSUMER\_SELECTOR, 281
  - CONSUMER\_MAX\_PENDING\_MSG\_LIMIT, 280
  - DESTINATION\_ADD\_OPERATION, 280
  - DESTINATION\_REMOVE\_OPERATION, 280
  - DestinationActions, 280
  - DestinationOption, 280
  - NUM\_OPTIONS, 281
  - NUM\_PARAMS, 281
  - PARAM\_CLIENTID, 281
  - PARAM\_PASSWORD, 281
  - PARAM\_USERNAME, 281
  - toDestinationOption, 282
  - toString, 282
  - toURIOption, 282
  - TRANSACTION\_STATE\_BEGIN, 281
  - TRANSACTION\_STATE\_COMMIT\_ON\_PHASE, 281
  - TRANSACTION\_STATE\_COMMIT\_TWOPHASE, 281
  - TRANSACTION\_STATE\_END, 281
  - TRANSACTION\_STATE\_FORGET, 281
  - TRANSACTION\_STATE\_PREPARE, 281
  - TRANSACTION\_STATE\_RECOVER, 281
  - TRANSACTION\_STATE\_ROLLBACK, 281
  - TransactionState, 281
  - URIParam, 281
- activemq::core::ActiveMQConstants::StaticInitializer, 3528
  - ~StaticInitializer, 3528
  - destOptionMap, 3529
  - destOptions, 3529
  - StaticInitializer, 3528
  - uriParams, 3529
  - uriParamsMap, 3529
- activemq::core::ActiveMQConsumer, 282
  - ~ActiveMQConsumer, 284
  - acknowledge, 284, 285
  - ActiveMQConsumer, 284
  - afterMessagesConsumed, 285
  - beforeMessagesConsumed, 285
  - clearMessagesInProgress, 285
  - close, 286
  - commit, 286
  - deliverAcks, 286
  - dequeue, 286
  - dispatch, 287
  - doClose, 287
  - getConsumerId, 287
  - getConsumerInfo, 287
  - getLastDeliveredSequenceId, 287
  - getMessageAvailableCount, 288
  - getMessageListener, 288
  - getMessageSelector, 288
  - getRedeliveryPolicy, 288
  - inProgressClearRequired, 289
  - isClosed, 289
  - isSynchronizationRegistered, 289
  - iterate, 289
  - receive, 289
  - receiveNoWait, 290
  - rollback, 290
  - setLastDeliveredSequenceId, 290
  - setMessageListener, 291
  - setRedeliveryPolicy, 291
  - setSynchronizationRegistered, 291
  - start, 291
  - stop, 291

- activemq::core::ActiveMQProducer, 441
  - ~ActiveMQProducer, 442
  - ActiveMQProducer, 442
  - close, 443
  - getDeliveryMode, 443
  - getDisableMessageID, 443
  - getDisableMessageTimeStamp, 443
  - getPriority, 443
  - getProducerId, 444
  - getProducerInfo, 444
  - getSendTimeout, 444
  - getTimeToLive, 444
  - isClosed, 444
  - onProducerAck, 445
  - send, 445–447
  - setDeliveryMode, 447
  - setDisableMessageID, 447
  - setDisableMessageTimeStamp, 448
  - setPriority, 448
  - setSendTimeout, 448
  - setTimeToLive, 448
- activemq::core::ActiveMQQueueBrowser, 457
  - ~ActiveMQQueueBrowser, 458
  - ActiveMQQueueBrowser, 458
  - Browser, 460
  - close, 458
  - getEnumeration, 458
  - getMessageSelector, 459
  - getQueue, 459
  - hasMoreMessages, 459
  - nextMessage, 459
- activemq::core::ActiveMQSession, 484
  - ~ActiveMQSession, 488
  - acknowledge, 488
  - ActiveMQSession, 488
  - ActiveMQSessionExecutor, 503
  - addConsumer, 488
  - addProducer, 488
  - clearMessagesInProgress, 489
  - close, 489
  - commit, 489
  - createBrowser, 489, 490
  - createBytesMessage, 490, 491
  - createConsumer, 491, 492
  - createDurableConsumer, 492
  - createMapMessage, 493
  - createMessage, 493
  - createProducer, 493
  - createQueue, 493
  - createStreamMessage, 494
  - createTemporaryQueue, 494
  - createTemporaryTopic, 494
  - createTextMessage, 495
  - createTopic, 495
  - deliverAcks, 496
  - dispatch, 496
  - doStartTransaction, 496
  - fire, 496
  - getAcknowledgeMode, 496
  - getConnection, 497
  - getExceptionListener, 497
  - getLastDeliveredSequenceId, 497
  - getNextConsumerId, 497
  - getNextProducerId, 497
  - getSessionId, 497
  - getSessionInfo, 498
  - getTransactionContext, 498
  - isAutoAcknowledge, 498
  - isClientAcknowledge, 498
  - isDupsOkAcknowledge, 498
  - isIndividualAcknowledge, 498
  - isStarted, 499
  - isTransacted, 499
  - oneway, 499
  - recover, 499
  - redispach, 500
  - removeConsumer, 500
  - removeProducer, 500
  - rollback, 501
  - send, 501
  - setLastDeliveredSequenceId, 501
  - start, 502
  - stop, 502
  - syncRequest, 502
  - unsubscribe, 502
  - wakeup, 502
- activemq::core::ActiveMQSessionExecutor, 503
  - ~ActiveMQSessionExecutor, 504
  - ActiveMQSessionExecutor, 504
  - clear, 504
  - clearMessagesInProgress, 504
  - close, 504
  - execute, 504
  - executeFirst, 505
  - getUnconsumedMessages, 505
  - hasUnconsumedMessages, 505
  - isEmpty, 505
  - isRunning, 505
  - iterate, 506

- start, 506
- stop, 506
- wakeup, 506
- activemq::core::ActiveMQTransactionContext, 688
  - ~ActiveMQTransactionContext, 689
  - ActiveMQTransactionContext, 689
  - addSynchronization, 689
  - begin, 689
  - commit, 689
  - getTransactionId, 690
  - isInTransaction, 690
  - removeSynchronization, 690
  - rollback, 690
- activemq::core::DispatchData, 1749
  - DispatchData, 1749
  - getConsumerId, 1749
  - getMessage, 1749
- activemq::core::Dispatcher, 1750
  - ~Dispatcher, 1750
  - dispatch, 1750
- activemq::core::MessageDispatchChannel, 2559
  - ~MessageDispatchChannel, 2561
  - clear, 2561
  - close, 2561
  - dequeue, 2561
  - dequeueNoWait, 2561
  - enqueue, 2561
  - enqueueFirst, 2562
  - isClosed, 2562
  - isEmpty, 2562
  - isRunning, 2562
  - lock, 2562
  - MessageDispatchChannel, 2561
  - notify, 2563
  - notifyAll, 2563
  - peek, 2563
  - removeAll, 2563
  - size, 2564
  - start, 2564
  - stop, 2564
  - tryLock, 2564
  - unlock, 2564
  - wait, 2565, 2566
- activemq::core::policies, 97
- activemq::core::policies::DefaultPrefetchPolicy, 1640
  - ~DefaultPrefetchPolicy, 1641
  - clone, 1641
  - DEFAULT\_DURABLE\_TOPIC\_PREFETCH, 1643
  - DEFAULT\_QUEUE\_BROWSER\_PREFETCH, 1644
  - DEFAULT\_QUEUE\_PREFETCH, 1644
  - DEFAULT\_TOPIC\_PREFETCH, 1644
  - DefaultPrefetchPolicy, 1641
  - getDurableTopicPrefetch, 1641
  - getMaxPrefetchLimit, 1641
  - getQueueBrowserPrefetch, 1642
  - getQueuePrefetch, 1642
  - getTopicPrefetch, 1642
  - MAX\_PREFETCH\_SIZE, 1644
  - setDurableTopicPrefetch, 1642
  - setQueueBrowserPrefetch, 1643
  - setQueuePrefetch, 1643
  - setTopicPrefetch, 1643
- activemq::core::policies::DefaultRedeliveryPolicy, 1644
  - ~DefaultRedeliveryPolicy, 1645
  - clone, 1645
  - DefaultRedeliveryPolicy, 1645
  - getBackOffMultiplier, 1645
  - getCollisionAvoidancePercent, 1645
  - getInitialRedeliveryDelay, 1645
  - getMaximumRedeliveries, 1646
  - getRedeliveryDelay, 1646
  - isUseCollisionAvoidance, 1646
  - isUseExponentialBackOff, 1646
  - setBackOffMultiplier, 1647
  - setCollisionAvoidancePercent, 1647
  - setInitialRedeliveryDelay, 1647
  - setMaximumRedeliveries, 1647
  - setUseCollisionAvoidance, 1648
  - setUseExponentialBackOff, 1648
- activemq::core::PrefetchPolicy, 2924
  - ~PrefetchPolicy, 2926
  - clone, 2926
  - configure, 2926
  - getDurableTopicPrefetch, 2926
  - getMaxPrefetchLimit, 2927
  - getQueueBrowserPrefetch, 2927
  - getQueuePrefetch, 2927
  - getTopicPrefetch, 2927
  - PrefetchPolicy, 2926
  - setDurableTopicPrefetch, 2927
  - setQueueBrowserPrefetch, 2928
  - setQueuePrefetch, 2928
  - setTopicPrefetch, 2928
- activemq::core::RedeliveryPolicy, 3121

- ~RedeliveryPolicy, 3123
- clone, 3123
- configure, 3123
- getBackOffMultiplier, 3123
- getCollisionAvoidancePercent, 3124
- getInitialRedeliveryDelay, 3124
- getMaximumRedeliveries, 3124
- getRedeliveryDelay, 3124
- isUseCollisionAvoidance, 3125
- isUseExponentialBackOff, 3125
- NO\_MAXIMUM\_REDELIVERIES, 3126
- RedeliveryPolicy, 3123
- setBackOffMultiplier, 3125
- setCollisionAvoidancePercent, 3125
- setInitialRedeliveryDelay, 3125
- setMaximumRedeliveries, 3126
- setUseCollisionAvoidance, 3126
- setUseExponentialBackOff, 3126
- activemq::core::Synchronization, 3659
  - ~Synchronization, 3659
  - afterCommit, 3659
  - afterRollback, 3659
  - beforeEnd, 3659
- activemq::exceptions, 97
- activemq::exceptions::ActiveMQException, 328
  - ~ActiveMQException, 329
  - ActiveMQException, 329
  - clone, 329
  - convertToCMSException, 330
- activemq::exceptions::BrokerException, 827
  - ~BrokerException, 828
  - BrokerException, 828
  - clone, 828
- activemq::io, 98
- activemq::io::LoggingInputStream, 2358
  - ~LoggingInputStream, 2358
  - doReadArrayBounded, 2358
  - doReadByte, 2359
  - LoggingInputStream, 2358
- activemq::io::LoggingOutputStream, 2359
  - ~LoggingOutputStream, 2360
  - doWriteArrayBounded, 2360
  - doWriteByte, 2360
  - LoggingOutputStream, 2360
- activemq::library, 98
- activemq::library::ActiveMQCPP, 292
  - ~ActiveMQCPP, 292
  - ActiveMQCPP, 292
  - initializeLibrary, 292, 293
  - operator=, 293
  - shutdownLibrary, 293
- activemq::state, 98
- activemq::state::CommandVisitor, 1171
  - ~CommandVisitor, 1173
  - processBeginTransaction, 1173
  - processBrokerError, 1173
  - processBrokerInfo, 1174
  - processCommitTransactionOnePhase, 1174
  - processCommitTransactionTwoPhase, 1174
  - processConnectionControl, 1174
  - processConnectionError, 1174
  - processConnectionInfo, 1174
  - processConsumerControl, 1174
  - processConsumerInfo, 1174
  - processControlCommand, 1175
  - processDestinationInfo, 1175
  - processEndTransaction, 1175
  - processFlushCommand, 1175
  - processForgetTransaction, 1175
  - processKeepAliveInfo, 1175
  - processMessage, 1175
  - processMessageAck, 1175
  - processMessageDispatch, 1176
  - processMessageDispatchNotification, 1176
  - processMessagePull, 1176
  - processPrepareTransaction, 1176
  - processProducerAck, 1176
  - processProducerInfo, 1176
  - processRecoverTransactions, 1176
  - processRemoveConnection, 1176
  - processRemoveConsumer, 1177
  - processRemoveDestination, 1177
  - processRemoveInfo, 1177
  - processRemoveProducer, 1177
  - processRemoveSession, 1177
  - processRemoveSubscriptionInfo, 1177
  - processReplayCommand, 1178
  - processResponse, 1178
  - processRollbackTransaction, 1178
  - processSessionInfo, 1178
  - processShutdownInfo, 1178
  - processTransactionInfo, 1178
  - processWireFormat, 1178
- activemq::state::CommandVisitorAdapter, 1179
  - ~CommandVisitorAdapter, 1181
  - processBeginTransaction, 1182

- processBrokerError, 1182
- processBrokerInfo, 1182
- processCommitTransactionOnePhase, 1182
- processCommitTransactionTwoPhase, 1182
- processConnectionControl, 1182
- processConnectionError, 1182
- processConnectionInfo, 1182
- processConsumerControl, 1182
- processConsumerInfo, 1183
- processControlCommand, 1183
- processDestinationInfo, 1183
- processEndTransaction, 1183
- processFlushCommand, 1183
- processForgetTransaction, 1183
- processKeepAliveInfo, 1183
- processMessage, 1183
- processMessageAck, 1183
- processMessageDispatch, 1184
- processMessageDispatchNotification, 1184
- processMessagePull, 1184
- processPrepareTransaction, 1184
- processProducerAck, 1184
- processProducerInfo, 1184
- processRecoverTransactions, 1184
- processRemoveConnection, 1184
- processRemoveConsumer, 1184
- processRemoveDestination, 1185
- processRemoveInfo, 1185
- processRemoveProducer, 1185
- processRemoveSession, 1185
- processRemoveSubscriptionInfo, 1185
- processReplayCommand, 1185
- processResponse, 1185
- processRollbackTransaction, 1185
- processSessionInfo, 1186
- processShutdownInfo, 1186
- processTransactionInfo, 1186
- processWireFormat, 1186
- activemq::state::ConnectionState, 1358
  - ~ConnectionState, 1359
  - addSession, 1359
  - addTempDestination, 1359
  - addTransactionState, 1359
  - checkShutdown, 1359
  - ConnectionState, 1359
  - getInfo, 1360
  - getRecoveringPullConsumers, 1360
  - getSessionState, 1360
  - getSessionStates, 1360
  - getTempDestinations, 1360
  - getTransactionState, 1360
  - getTransactionStates, 1360
  - isConnectionInterruptProcessingComplete, 1360
  - removeSession, 1360
  - removeTempDestination, 1360
  - removeTransactionState, 1360
  - reset, 1360
  - setConnectionInterruptProcessingComplete, 1360
  - shutdown, 1361
  - toString, 1361
- activemq::state::ConnectionStateTracker, 1361
  - ~ConnectionStateTracker, 1362
  - connectionInterruptProcessingComplete, 1363
  - ConnectionStateTracker, 1362
  - getMaxCacheSize, 1363
  - isRestoreConsumers, 1363
  - isRestoreProducers, 1363
  - isRestoreSessions, 1363
  - isRestoreTransaction, 1363
  - isTrackMessages, 1363
  - isTrackTransactionProducers, 1363
  - isTrackTransactions, 1363
  - processBeginTransaction, 1363
  - processCommitTransactionOnePhase, 1363
  - processCommitTransactionTwoPhase, 1363
  - processConnectionInfo, 1364
  - processConsumerInfo, 1364
  - processDestinationInfo, 1364
  - processEndTransaction, 1364
  - processMessage, 1364
  - processMessageAck, 1364
  - processPrepareTransaction, 1365
  - processProducerInfo, 1365
  - processRemoveConnection, 1365
  - processRemoveConsumer, 1365
  - processRemoveDestination, 1365
  - processRemoveProducer, 1365
  - processRemoveSession, 1366
  - processRollbackTransaction, 1366
  - processSessionInfo, 1366
  - RemoveTransactionAction, 1367
  - restore, 1366



- setMaxCacheSize, 1366
- setRestoreConsumers, 1366
- setRestoreProducers, 1366
- setRestoreSessions, 1366
- setRestoreTransaction, 1366
- setTrackMessages, 1366
- setTrackTransactionProducers, 1367
- setTrackTransactions, 1367
- track, 1367
- trackBack, 1367
- transportInterrupted, 1367
- activemq::state::ConsumerState, 1459
  - ~ConsumerState, 1459
  - ConsumerState, 1459
  - getInfo, 1459
  - toString, 1459
- activemq::state::ProducerState, 3072
  - ~ProducerState, 3072
  - getInfo, 3072
  - getTransactionState, 3072
  - ProducerState, 3072
  - setTransactionState, 3072
  - toString, 3072
- activemq::state::SessionState, 3378
  - ~SessionState, 3378
  - addConsumer, 3378
  - addProducer, 3378
  - checkShutdown, 3378
  - getConsumerState, 3379
  - getConsumerStates, 3379
  - getInfo, 3379
  - getProducerState, 3379
  - getProducerStates, 3379
  - removeConsumer, 3379
  - removeProducer, 3379
  - SessionState, 3378
  - shutdown, 3379
  - toString, 3379
- activemq::state::Tracked, 3758
  - ~Tracked, 3759
  - isWaitingForResponse, 3759
  - onResponse, 3759
  - Tracked, 3759
- activemq::state::TransactionState, 3813
  - ~TransactionState, 3814
  - addCommand, 3814
  - addProducerState, 3814
  - checkShutdown, 3814
  - getCommands, 3814
  - getId, 3814
  - getPreparedResult, 3814
  - getProducerStates, 3814
  - isPrepared, 3814
  - setPrepared, 3814
  - setPreparedResult, 3814
  - shutdown, 3814
  - toString, 3814
  - TransactionState, 3814
- activemq::threads, 98
- activemq::threads::CompositeTask, 1193
  - ~CompositeTask, 1193
  - isPending, 1193
- activemq::threads::CompositeTaskRunner, 1194
  - ~CompositeTaskRunner, 1195
  - addTask, 1195
  - CompositeTaskRunner, 1195
  - iterate, 1195
  - removeTask, 1195
  - run, 1196
  - shutdown, 1196
  - wakeup, 1196
- activemq::threads::DedicatedTaskRunner, 1638
  - ~DedicatedTaskRunner, 1639
  - DedicatedTaskRunner, 1639
  - run, 1639
  - shutdown, 1639
  - wakeup, 1640
- activemq::threads::Task, 3678
  - ~Task, 3679
  - iterate, 3679
- activemq::threads::TaskRunner, 3680
  - ~TaskRunner, 3681
  - shutdown, 3681
  - wakeup, 3681
- activemq::transport, 99
- activemq::transport::AbstractTransportFactory, 170
  - ~AbstractTransportFactory, 171
  - createWireFormat, 171
- activemq::transport::CompositeTransport, 1197
  - ~CompositeTransport, 1197
  - addURI, 1197
  - removeURI, 1198
- activemq::transport::correlator, 100
- activemq::transport::correlator::FutureResponse, 1932
  - ~FutureResponse, 1933
  - FutureResponse, 1933
  - getResponse, 1933

- setResponse, 1934
- activemq::transport::correlator::ResponseCorrelator, 1835
  - 3232
  - ~ResponseCorrelator, 3234
  - close, 3234
  - onCommand, 3234
  - oneway, 3234
  - onTransportException, 3235
  - request, 3235
  - ResponseCorrelator, 3233
  - start, 3236
- activemq::transport::DefaultTransportListener, 1670
  - ~DefaultTransportListener, 1671
  - onCommand, 1671
  - onException, 1671
  - transportInterrupted, 1671
  - transportResumed, 1671
- activemq::transport::failover, 100
- activemq::transport::failover::BackupTransport, 718
  - ~BackupTransport, 719
  - BackupTransport, 719
  - getTransport, 719
  - getUri, 719
  - isClosed, 719
  - onException, 719
  - setClosed, 720
  - setTransport, 720
  - setUri, 720
- activemq::transport::failover::BackupTransportPool, 720
  - ~BackupTransportPool, 721
  - BackupTransport, 723
  - BackupTransportPool, 721
  - getBackup, 721
  - getBackupPoolSize, 722
  - isEnabled, 722
  - isPending, 722
  - iterate, 722
  - setBackupPoolSize, 722
  - setEnabled, 723
- activemq::transport::failover::CloseTransportsTask, 1122
  - ~CloseTransportsTask, 1122
  - add, 1122
  - CloseTransportsTask, 1122
  - isPending, 1122
  - iterate, 1122
- activemq::transport::failover::FailoverTransport, 1835
  - ~FailoverTransport, 1837
  - add, 1837
  - addURI, 1838
  - close, 1838
  - FailoverTransport, 1837
  - FailoverTransportListener, 1846
  - getBackOffMultiplier, 1838
  - getBackupPoolSize, 1838
  - getInitialReconnectDelay, 1838
  - getMaxCacheSize, 1838
  - getMaxReconnectAttempts, 1838
  - getMaxReconnectDelay, 1839
  - getReconnectDelay, 1839
  - getRemoteAddress, 1839
  - getStartupMaxReconnectAttempts, 1839
  - getTimeout, 1839
  - getTransportListener, 1839
  - handleTransportFailure, 1839
  - isBackup, 1840
  - isClosed, 1840
  - isConnected, 1840
  - isFaultTolerant, 1840
  - isInitialized, 1840
  - isPending, 1840
  - isRandomize, 1841
  - isTrackMessages, 1841
  - isTrackTransactionProducers, 1841
  - isUseExponentialBackOff, 1841
  - iterate, 1841
  - narrow, 1841
  - oneway, 1842
  - reconnect, 1842
  - removeURI, 1842
  - request, 1843
  - restoreTransport, 1844
  - setBackOffMultiplier, 1844
  - setBackup, 1844
  - setBackupPoolSize, 1844
  - setConnectionInterruptProcessingComplete, 1844
  - setInitialized, 1844
  - setInitialReconnectDelay, 1844
  - setMaxCacheSize, 1844
  - setMaxReconnectAttempts, 1845
  - setMaxReconnectDelay, 1845
  - setRandomize, 1845
  - setReconnectDelay, 1845
  - setStartupMaxReconnectAttempts, 1845

- setTimeout, 1845
- setTrackMessages, 1845
- setTrackTransactionProducers, 1845
- setTransportListener, 1845
- setUseExponentialBackOff, 1845
- setWireFormat, 1845
- start, 1846
- stop, 1846
- activemq::transport::failover::FailoverTransportFactory, 3950
  - 1846
  - ~FailoverTransportFactory, 1847
  - create, 1847
  - createComposite, 1848
  - doCreateComposite, 1848
- activemq::transport::failover::FailoverTransportListener, 1849
  - ~FailoverTransportListener, 1849
  - FailoverTransportListener, 1849
  - onCommand, 1849
  - onException, 1850
  - transportInterrupted, 1850
  - transportResumed, 1850
- activemq::transport::failover::URIPool, 3875
  - ~URIPool, 3875
  - addURI, 3876
  - addURIs, 3876
  - getURI, 3876
  - isRandomize, 3876
  - removeURI, 3877
  - setRandomize, 3877
  - URIPool, 3875
- activemq::transport::inactivity, 100
- activemq::transport::inactivity::InactivityMonitor, 1964
  - ~InactivityMonitor, 1965
  - AsyncSignalReadErrorTask, 1967
  - AsyncWriteTask, 1967
  - close, 1965
  - getInitialDelayTime, 1966
  - getReadCheckTime, 1966
  - getWriteCheckTime, 1966
  - InactivityMonitor, 1965
  - isKeepAliveResponseRequired, 1966
  - onCommand, 1966
  - oneway, 1966
  - onException, 1966
  - ReadChecker, 1967
  - setInitialDelayTime, 1967
  - setKeepAliveResponseRequired, 1967
  - setReadCheckTime, 1967
  - setWriteCheckTime, 1967
  - WriteChecker, 1967
- activemq::transport::inactivity::ReadChecker, 3107
  - ~ReadChecker, 3108
  - ReadChecker, 3108
  - run, 3108
- activemq::transport::inactivity::WriteChecker, 3951
  - ~WriteChecker, 3951
  - run, 3951
  - WriteChecker, 3951
- activemq::transport::IOTransport, 2105
  - ~IOTransport, 2107
  - close, 2107
  - getRemoteAddress, 2108
  - getTransportListener, 2108
  - IOTransport, 2107
  - isClosed, 2108
  - isConnected, 2108
  - isFaultTolerant, 2108
  - narrow, 2109
  - oneway, 2109
  - reconnect, 2109
  - request, 2110
  - run, 2111
  - setInputStream, 2111
  - setOutputStream, 2111
  - setTransportListener, 2111
  - setWireFormat, 2111
  - start, 2112
  - stop, 2112
- activemq::transport::logging, 100
- activemq::transport::logging::LoggingTransport, 2360
  - ~LoggingTransport, 2361
  - LoggingTransport, 2361
  - onCommand, 2361
  - oneway, 2362
  - request, 2362
- activemq::transport::mock, 101
- activemq::transport::mock::InternalCommandListener, 2085
  - ~InternalCommandListener, 2085
  - InternalCommandListener, 2085
  - onCommand, 2086
  - run, 2086
  - setResponseBuilder, 2086
  - setTransport, 2086

- activemq::transport::mock::MockTransport, 2724
  - ~MockTransport, 2726
  - close, 2726
  - fireCommand, 2726
  - fireException, 2727
  - getInstance, 2727
  - getNumReceivedMessageBeforeFail, 2727
  - getNumReceivedMessages, 2727
  - getNumSentKeepAlives, 2727
  - getNumSentKeepAlivesBeforeFail, 2727
  - getNumSentMessageBeforeFail, 2727
  - getNumSentMessages, 2727
  - getRemoteAddress, 2727
  - getTransportListener, 2727
  - getWireFormat, 2728
  - isClosed, 2728
  - isConnected, 2728
  - isFailOnClose, 2728
  - isFailOnKeepAliveSends, 2729
  - isFailOnReceiveMessage, 2729
  - isFailOnSendMessage, 2729
  - isFailOnStart, 2729
  - isFailOnStop, 2729
  - isFaultTolerant, 2729
  - MockTransport, 2726
  - narrow, 2729
  - oneway, 2729
  - reconnect, 2730
  - request, 2730, 2731
  - setFailOnClose, 2731
  - setFailOnKeepAliveSends, 2731
  - setFailOnReceiveMessage, 2731
  - setFailOnSendMessage, 2731
  - setFailOnStart, 2731
  - setFailOnStop, 2731
  - setNumReceivedMessageBeforeFail, 2731
  - setNumReceivedMessages, 2732
  - setNumSentKeepAlives, 2732
  - setNumSentKeepAlivesBeforeFail, 2732
  - setNumSentMessageBeforeFail, 2732
  - setNumSentMessages, 2732
  - setOutgoingListener, 2732
  - setResponseBuilder, 2732
  - setTransportListener, 2732
  - setWireFormat, 2733
  - start, 2733
  - stop, 2733
- activemq::transport::mock::MockTransportFactory, 2734
  - ~MockTransportFactory, 2734
  - create, 2734
  - createComposite, 2735
  - doCreateComposite, 2735
- activemq::transport::mock::ResponseBuilder, 3231
  - ~ResponseBuilder, 3232
  - buildIncomingCommands, 3232
  - buildResponse, 3232
- activemq::transport::tcp, 101
- activemq::transport::tcp::SslTransport, 3518
  - ~SslTransport, 3519
  - configureSocket, 3519
  - createSocket, 3519
  - SslTransport, 3519
- activemq::transport::tcp::SslTransportFactory, 3520
  - ~SslTransportFactory, 3520
  - doCreateComposite, 3520
- activemq::transport::tcp::TcpTransport, 3696
  - ~TcpTransport, 3697
  - close, 3697
  - configureSocket, 3697
  - connect, 3698
  - createSocket, 3698
  - isClosed, 3698
  - isConnected, 3699
  - isFaultTolerant, 3699
  - TcpTransport, 3697
- activemq::transport::tcp::TcpTransportFactory, 3699
  - ~TcpTransportFactory, 3700
  - create, 3700
  - createComposite, 3700
  - doCreateComposite, 3701
- activemq::transport::Transport, 3819
  - ~Transport, 3820
  - getRemoteAddress, 3820
  - getTransportListener, 3821
  - isClosed, 3821
  - isConnected, 3821
  - isFaultTolerant, 3821
  - narrow, 3822
  - oneway, 3822
  - reconnect, 3823
  - request, 3823, 3824
  - setTransportListener, 3824
  - setWireFormat, 3824

- start, 3825
- stop, 3825
- activemq::transport::TransportFactory, 3825
  - ~TransportFactory, 3826
  - create, 3826
  - createComposite, 3826
- activemq::transport::TransportFilter, 3827
  - ~TransportFilter, 3829
  - close, 3829
  - fire, 3830
  - getRemoteAddress, 3830
  - getTransportListener, 3830
  - isClosed, 3830
  - isConnected, 3831
  - isFaultTolerant, 3831
  - listener, 3835
  - narrow, 3831
  - next, 3835
  - onCommand, 3831
  - oneway, 3832
  - onException, 3832
  - reconnect, 3833
  - request, 3833
  - setTransportListener, 3834
  - setWireFormat, 3834
  - start, 3834
  - stop, 3835
  - TransportFilter, 3829
  - transportInterrupted, 3835
  - transportResumed, 3835
- activemq::transport::TransportListener, 3836
  - ~TransportListener, 3836
  - onCommand, 3836
  - onException, 3837
  - transportInterrupted, 3837
  - transportResumed, 3837
- activemq::transport::TransportRegistry, 3837
  - ~TransportRegistry, 3838
  - findFactory, 3838
  - getInstance, 3839
  - getTransportNames, 3839
  - registerFactory, 3839
  - unregisterFactory, 3840
- activemq::util, 101
- activemq::util::ActiveMQProperties, 449
  - ~ActiveMQProperties, 450
  - ActiveMQProperties, 450
  - clear, 450
  - clone, 450
  - copy, 450
  - getProperties, 450
  - getProperty, 451
  - hasProperty, 451
  - isEmpty, 451
  - remove, 452
  - setProperties, 452
  - setProperty, 452
  - toArray, 452
  - toString, 452
- activemq::util::CMSExceptionSupport, 1134
  - ~CMSExceptionSupport, 1134
  - create, 1134
  - createMessageEOFException, 1134
  - createMessageFormatException, 1134
- activemq::util::CompositeData, 1191
  - ~CompositeData, 1192
  - CompositeData, 1192
  - getComponents, 1192
  - getFragment, 1192
  - getHost, 1192
  - getParameters, 1192
  - getPath, 1192
  - getScheme, 1192
  - setComponents, 1192
  - setFragment, 1192
  - setHost, 1192
  - setParameters, 1192
  - setPath, 1192
  - setScheme, 1192
  - toURI, 1193
- activemq::util::IdGenerator, 1951
  - ~IdGenerator, 1951
  - compare, 1951
  - generateId, 1952
  - getHostname, 1952
  - getSeedFromId, 1952
  - getSequenceFromId, 1952
  - IdGenerator, 1951
- activemq::util::LongSequenceGenerator, 2415
  - ~LongSequenceGenerator, 2416
  - getLastSequenceId, 2416
  - getNextSequenceId, 2416
  - LongSequenceGenerator, 2416
- activemq::util::MarshallingSupport, 2451
  - ~MarshallingSupport, 2452
  - asciiToModifiedUtf8, 2452
  - MarshallingSupport, 2452
  - modifiedUtf8ToAscii, 2452
  - readString16, 2453
  - readString32, 2453

- writeString, 2454
- writeString16, 2454
- writeString32, 2455
- activemq::util::MemoryUsage, 2472
  - ~MemoryUsage, 2473
  - decreaseUsage, 2473
  - enqueueUsage, 2473
  - getLimit, 2474
  - getUsage, 2474
  - increaseUsage, 2474
  - isFull, 2474
  - MemoryUsage, 2473
  - setLimit, 2474
  - setUsage, 2475
  - waitForSpace, 2475
- activemq::util::PrimitiveList, 2929
  - ~PrimitiveList, 2931
  - getBool, 2931
  - getByte, 2932
  - getByteArray, 2932
  - getChar, 2933
  - getDouble, 2933
  - getFloat, 2934
  - getInt, 2934
  - getLong, 2935
  - getShort, 2935
  - getString, 2936
  - PrimitiveList, 2931
  - setBool, 2936
  - setByte, 2937
  - setByteArray, 2937
  - setChar, 2938
  - setDouble, 2938
  - setFloat, 2938
  - setInt, 2939
  - setLong, 2939
  - setShort, 2939
  - setString, 2940
  - toString, 2940
- activemq::util::PrimitiveMap, 2941
  - ~PrimitiveMap, 2943
  - getBool, 2943
  - getByte, 2944
  - getByteArray, 2944
  - getChar, 2945
  - getDouble, 2945
  - getFloat, 2946
  - getInt, 2946
  - getLong, 2946
  - getShort, 2947
  - getString, 2947
  - PrimitiveMap, 2943
  - setBool, 2948
  - setByte, 2948
  - setByteArray, 2948
  - setChar, 2949
  - setDouble, 2949
  - setFloat, 2949
  - setInt, 2949
  - setLong, 2950
  - setShort, 2950
  - setString, 2950
  - toString, 2950
- activemq::util::PrimitiveValueConverter, 2959
  - ~PrimitiveValueConverter, 2959
  - convert, 2960
  - PrimitiveValueConverter, 2959
- activemq::util::PrimitiveValueNode, 2960
  - ~PrimitiveValueNode, 2967
  - BIG\_STRING\_TYPE, 2964
  - BOOLEAN\_TYPE, 2963
  - BYTE\_ARRAY\_TYPE, 2964
  - BYTE\_TYPE, 2963
  - CHAR\_TYPE, 2963
  - clear, 2967
  - DOUBLE\_TYPE, 2964
  - FLOAT\_TYPE, 2964
  - getBool, 2967
  - getByte, 2967
  - getByteArray, 2967
  - getChar, 2968
  - getDouble, 2968
  - getFloat, 2968
  - getInt, 2969
  - getList, 2969
  - getLong, 2969
  - getMap, 2969
  - getShort, 2970
  - getString, 2970
  - getType, 2970
  - getValue, 2971
  - INTEGER\_TYPE, 2963
  - LIST\_TYPE, 2964
  - LONG\_TYPE, 2964
  - MAP\_TYPE, 2964
  - NULL\_TYPE, 2963
  - operator=, 2971
  - operator==, 2971
  - PrimitiveType, 2963
  - PrimitiveValueNode, 2964–2966

- setBool, 2971
- setByte, 2971
- setByteArray, 2972
- setChar, 2972
- setDouble, 2972
- setFloat, 2972
- setInt, 2972
- setList, 2973
- setLong, 2973
- setMap, 2973
- setShort, 2973
- setString, 2974
- setValue, 2974
- SHORT\_TYPE, 2963
- STRING\_TYPE, 2964
- toString, 2974
- activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2957
  - boolValue, 2958
  - byteArrayValue, 2958
  - byteValue, 2958
  - charValue, 2958
  - doubleValue, 2958
  - floatValue, 2958
  - intValue, 2958
  - listValue, 2958
  - longValue, 2958
  - mapValue, 2958
  - shortValue, 2958
  - stringValue, 2958
- activemq::util::URISupport, 3877
  - createQueryString, 3878
  - parseComposite, 3878
  - parseQuery, 3878, 3879
  - parseURL, 3879
- activemq::util::Usage, 3895
  - ~Usage, 3896
  - decreaseUsage, 3896
  - enqueueUsage, 3896
  - increaseUsage, 3896
  - isFull, 3896
  - waitForSpace, 3897
- activemq::wireformat, 102
- activemq::wireformat::MarshalAware, 2444
  - ~MarshalAware, 2445
  - afterMarshal, 2445
  - afterUnmarshal, 2445
  - beforeMarshal, 2445
  - beforeUnmarshal, 2445
  - getMarshaledForm, 2445
  - isMarshalAware, 2446
  - setMarshaledForm, 2446
- activemq::wireformat::openwire, 102
- activemq::wireformat::openwire::marshal, 103
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 770
  - ~BaseDataStreamMarshaller, 776
  - looseMarshal, 776
  - looseMarshalBrokerError, 776
  - looseMarshalCachedObject, 777
  - looseMarshalLong, 777
  - looseMarshalNestedObject, 777
  - looseMarshalObjectArray, 778
  - looseMarshalString, 778
  - looseUnmarshal, 778
  - looseUnmarshalBrokerError, 779
  - looseUnmarshalByteArray, 779
  - looseUnmarshalCachedObject, 780
  - looseUnmarshalConstByteArray, 780
  - looseUnmarshalLong, 780
  - looseUnmarshalNestedObject, 781
  - looseUnmarshalString, 781
  - readAsciiString, 782
  - tightMarshal1, 782
  - tightMarshal2, 782
  - tightMarshalBrokerError1, 783
  - tightMarshalBrokerError2, 783
  - tightMarshalCachedObject1, 783
  - tightMarshalCachedObject2, 784
  - tightMarshalLong1, 784
  - tightMarshalLong2, 785
  - tightMarshalNestedObject1, 785
  - tightMarshalNestedObject2, 786
  - tightMarshalObjectArray1, 786
  - tightMarshalObjectArray2, 786
  - tightMarshalString1, 787
  - tightMarshalString2, 787
  - tightUnmarshal, 788
  - tightUnmarshalBrokerError, 788
  - tightUnmarshalByteArray, 789
  - tightUnmarshalCachedObject, 789
  - tightUnmarshalConstByteArray, 789
  - tightUnmarshalLong, 790
  - tightUnmarshalNestedObject, 790
  - tightUnmarshalString, 791
  - toHexFromBytes, 791
  - toString, 792
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1577
  - ~DataStreamMarshaller, 1578

- createObject, 1578
- getDataStructureType, 1585
- looseMarshal, 1591
- looseUnmarshal, 1598
- tightMarshal1, 1606
- tightMarshal2, 1613
- tightUnmarshal, 1620
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2951
  - ~PrimitiveTypesMarshaller, 2952
  - marshal, 2953
  - marshalList, 2953
  - marshalMap, 2954
  - marshalPrimitive, 2954
  - marshalPrimitiveList, 2954
  - marshalPrimitiveMap, 2954
  - PrimitiveTypesMarshaller, 2952
  - unmarshal, 2955
  - unmarshalList, 2955
  - unmarshalMap, 2956
  - unmarshalPrimitive, 2956
  - unmarshalPrimitiveList, 2957
  - unmarshalPrimitiveMap, 2957
- activemq::wireformat::openwire::marshal::v1, 103
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 182
  - ~ActiveMQBlobMessageMarshaller, 183
  - ActiveMQBlobMessageMarshaller, 183
  - createObject, 183
  - getDataStructureType, 183
  - looseMarshal, 183
  - looseUnmarshal, 184
  - tightMarshal1, 184
  - tightMarshal2, 185
  - tightUnmarshal, 185
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 224
  - ~ActiveMQBytesMessageMarshaller, 225
  - ActiveMQBytesMessageMarshaller, 225
  - createObject, 225
  - getDataStructureType, 226
  - looseMarshal, 226
  - looseUnmarshal, 226
  - tightMarshal1, 227
  - tightMarshal2, 227
  - tightUnmarshal, 228
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 308
  - ~ActiveMQDestinationMarshaller, 309
  - ActiveMQDestinationMarshaller, 309
  - looseMarshal, 309
  - looseUnmarshal, 310
  - tightMarshal1, 310
  - tightMarshal2, 311
  - tightUnmarshal, 311
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 348
  - ~ActiveMQMapMessageMarshaller, 349
  - ActiveMQMapMessageMarshaller, 349
  - createObject, 349
  - getDataStructureType, 350
  - looseMarshal, 350
  - looseUnmarshal, 350
  - tightMarshal1, 351
  - tightMarshal2, 351
  - tightUnmarshal, 352
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 375
  - ~ActiveMQMessageMarshaller, 376
  - ActiveMQMessageMarshaller, 376
  - createObject, 376
  - getDataStructureType, 376
  - looseMarshal, 376
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 421
  - ~ActiveMQObjectMessageMarshaller, 422
  - ActiveMQObjectMessageMarshaller, 422
  - createObject, 422
  - getDataStructureType, 422
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 464
  - ~ActiveMQQueueMarshaller, 465
  - ActiveMQQueueMarshaller, 465
  - createObject, 465
  - getDataStructureType, 465
  - looseMarshal, 466
  - tightMarshal1, 467



- tightMarshal2, 467
- tightUnmarshal, 467
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 527
- ~ActiveMQStreamMessageMarshaller, 528
- ActiveMQStreamMessageMarshaller, 528
- createObject, 528
- getDataStructureType, 528
- looseMarshal, 529
- looseUnmarshal, 529
- tightMarshal1, 530
- tightMarshal2, 530
- tightUnmarshal, 530
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 555
- ~ActiveMQTempDestinationMarshaller, 556
- ActiveMQTempDestinationMarshaller, 556
- looseMarshal, 556
- looseUnmarshal, 556
- tightMarshal1, 557
- tightMarshal2, 557
- tightUnmarshal, 558
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 582
- ~ActiveMQTempQueueMarshaller, 583
- ActiveMQTempQueueMarshaller, 583
- createObject, 583
- getDataStructureType, 584
- looseMarshal, 584
- looseUnmarshal, 584
- tightMarshal1, 585
- tightMarshal2, 585
- tightUnmarshal, 586
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 615
- ~ActiveMQTempTopicMarshaller, 616
- ActiveMQTempTopicMarshaller, 616
- createObject, 616
- getDataStructureType, 616
- looseMarshal, 616
- looseUnmarshal, 617
- tightMarshal1, 617
- tightMarshal2, 618
- tightUnmarshal, 618
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 644
- ~ActiveMQTextMessageMarshaller, 645
- ActiveMQTextMessageMarshaller, 645
- createObject, 645
- getDataStructureType, 645
- looseMarshal, 645
- looseUnmarshal, 646
- tightMarshal1, 646
- tightMarshal2, 647
- tightUnmarshal, 647
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 672
- ~ActiveMQTopicMarshaller, 673
- ActiveMQTopicMarshaller, 673
- createObject, 673
- getDataStructureType, 673
- looseMarshal, 673
- looseUnmarshal, 674
- tightMarshal1, 674
- tightMarshal2, 675
- tightUnmarshal, 675
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 743
- ~BaseCommandMarshaller, 744
- BaseCommandMarshaller, 744
- looseMarshal, 744
- looseUnmarshal, 746
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 840
- ~BrokerIdMarshaller, 841
- BrokerIdMarshaller, 841
- createObject, 841
- getDataStructureType, 841
- looseMarshal, 841
- looseUnmarshal, 842
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 871
- ~BrokerInfoMarshaller, 872
- BrokerInfoMarshaller, 872
- createObject, 872
- getDataStructureType, 872
- looseMarshal, 872
- looseUnmarshal, 873
- activemq::wireformat::openwire::marshal::v1::BrokerNameMarshaller, 879
- ~BrokerNameMarshaller, 880
- BrokerNameMarshaller, 880
- createObject, 880
- getDataStructureType, 880
- looseMarshal, 880
- looseUnmarshal, 881
- activemq::wireformat::openwire::marshal::v1::BrokerUrlMarshaller, 885
- ~BrokerUrlMarshaller, 886
- BrokerUrlMarshaller, 886
- createObject, 886
- getDataStructureType, 886
- looseMarshal, 886
- looseUnmarshal, 887
- activemq::wireformat::openwire::marshal::v1::BrokerVersionMarshaller, 891
- ~BrokerVersionMarshaller, 892
- BrokerVersionMarshaller, 892
- createObject, 892
- getDataStructureType, 892
- looseMarshal, 892
- looseUnmarshal, 893
- activemq::wireformat::openwire::marshal::v1::BrokerWeightMarshaller, 897
- ~BrokerWeightMarshaller, 898
- BrokerWeightMarshaller, 898
- createObject, 898
- getDataStructureType, 898
- looseMarshal, 898
- looseUnmarshal, 899
- activemq::wireformat::openwire::marshal::v1::BrokerXidMarshaller, 903
- ~BrokerXidMarshaller, 904
- BrokerXidMarshaller, 904
- createObject, 904
- getDataStructureType, 904
- looseMarshal, 904
- looseUnmarshal, 905
- activemq::wireformat::openwire::marshal::v1::BrokerYidMarshaller, 909
- ~BrokerYidMarshaller, 910
- BrokerYidMarshaller, 910
- createObject, 910
- getDataStructureType, 910
- looseMarshal, 910
- looseUnmarshal, 911
- activemq::wireformat::openwire::marshal::v1::BrokerZidMarshaller, 915
- ~BrokerZidMarshaller, 916
- BrokerZidMarshaller, 916
- createObject, 916
- getDataStructureType, 916
- looseMarshal, 916
- looseUnmarshal, 917
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 921
- ~BrokerZxidMarshaller, 922
- BrokerZxidMarshaller, 922
- createObject, 922
- getDataStructureType, 922
- looseMarshal, 922
- looseUnmarshal, 923
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 927
- ~BrokerZxidMarshaller, 928
- BrokerZxidMarshaller, 928
- createObject, 928
- getDataStructureType, 928
- looseMarshal, 928
- looseUnmarshal, 929
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 933
- ~BrokerZxidMarshaller, 934
- BrokerZxidMarshaller, 934
- createObject, 934
- getDataStructureType, 934
- looseMarshal, 934
- looseUnmarshal, 935
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 939
- ~BrokerZxidMarshaller, 940
- BrokerZxidMarshaller, 940
- createObject, 940
- getDataStructureType, 940
- looseMarshal, 940
- looseUnmarshal, 941
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 945
- ~BrokerZxidMarshaller, 946
- BrokerZxidMarshaller, 946
- createObject, 946
- getDataStructureType, 946
- looseMarshal, 946
- looseUnmarshal, 947
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 951
- ~BrokerZxidMarshaller, 952
- BrokerZxidMarshaller, 952
- createObject, 952
- getDataStructureType, 952
- looseMarshal, 952
- looseUnmarshal, 953
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 957
- ~BrokerZxidMarshaller, 958
- BrokerZxidMarshaller, 958
- createObject, 958
- getDataStructureType, 958
- looseMarshal, 958
- looseUnmarshal, 959
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 963
- ~BrokerZxidMarshaller, 964
- BrokerZxidMarshaller, 964
- createObject, 964
- getDataStructureType, 964
- looseMarshal, 964
- looseUnmarshal, 965
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 969
- ~BrokerZxidMarshaller, 970
- BrokerZxidMarshaller, 970
- createObject, 970
- getDataStructureType, 970
- looseMarshal, 970
- looseUnmarshal, 971
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 975
- ~BrokerZxidMarshaller, 976
- BrokerZxidMarshaller, 976
- createObject, 976
- getDataStructureType, 976
- looseMarshal, 976
- looseUnmarshal, 977
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 981
- ~BrokerZxidMarshaller, 982
- BrokerZxidMarshaller, 982
- createObject, 982
- getDataStructureType, 982
- looseMarshal, 982
- looseUnmarshal, 983
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 987
- ~BrokerZxidMarshaller, 988
- BrokerZxidMarshaller, 988
- createObject, 988
- getDataStructureType, 988
- looseMarshal, 988
- looseUnmarshal, 989
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 993
- ~BrokerZxidMarshaller, 994
- BrokerZxidMarshaller, 994
- createObject, 994
- getDataStructureType, 994
- looseMarshal, 994
- looseUnmarshal, 995
- activemq::wireformat::openwire::marshal::v1::BrokerZxidMarshaller, 999
- ~BrokerZxidMarshaller, 1000
- BrokerZxidMarshaller, 1000
- createObject, 1000
- getDataStructureType, 1000
- looseMarshal, 1000
- looseUnmarshal, 1001

- tightUnmarshal, 874
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1250
  - ~ControlCommandMarshaller, 1251
  - ControlCommandMarshaller, 1251
  - createObject, 1251
  - getDataSetType, 1251
  - looseMarshal, 1251
  - looseUnmarshal, 1252
  - tightMarshal1, 1252
  - tightMarshal2, 1253
  - tightUnmarshal, 1253
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1282
  - ~ConnectionErrorMarshaller, 1283
  - ConnectionErrorMarshaller, 1283
  - createObject, 1283
  - getDataSetType, 1283
  - looseMarshal, 1283
  - looseUnmarshal, 1284
  - tightMarshal1, 1284
  - tightMarshal2, 1285
  - tightUnmarshal, 1285
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1313
  - ~ConsumerIdMarshaller, 1314
  - ConsumerIdMarshaller, 1314
  - createObject, 1314
  - getDataSetType, 1314
  - looseMarshal, 1314
  - looseUnmarshal, 1315
  - tightMarshal1, 1315
  - tightMarshal2, 1315
  - tightUnmarshal, 1316
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1343
  - ~ConsumerInfoMarshaller, 1344
  - ConsumerInfoMarshaller, 1344
  - createObject, 1344
  - getDataSetType, 1344
  - looseMarshal, 1344
  - looseUnmarshal, 1345
  - tightMarshal1, 1345
  - tightMarshal2, 1346
  - tightUnmarshal, 1346
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1386
  - ~ConsumerControlMarshaller, 1387
  - ConsumerControlMarshaller, 1387
  - createObject, 1387
  - getDataSetType, 1387
  - looseMarshal, 1387
  - looseUnmarshal, 1388
  - tightMarshal1, 1388
  - tightMarshal2, 1389
  - tightUnmarshal, 1389
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1414
  - ~ConsumerIdMarshaller, 1415
  - ConsumerIdMarshaller, 1415
  - createObject, 1415
  - getDataSetType, 1415
  - looseMarshal, 1415
  - looseUnmarshal, 1416
  - tightMarshal1, 1416
  - tightMarshal2, 1417
  - tightUnmarshal, 1417
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1447
  - ~ConsumerInfoMarshaller, 1448
  - ConsumerInfoMarshaller, 1448
  - createObject, 1448
  - getDataSetType, 1448
  - looseMarshal, 1448
  - looseUnmarshal, 1449
  - tightMarshal1, 1449
  - tightMarshal2, 1450
  - tightUnmarshal, 1450
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1475
  - ~ControlCommandMarshaller, 1476
  - ControlCommandMarshaller, 1476
  - createObject, 1476
  - getDataSetType, 1476
  - looseMarshal, 1476
  - looseUnmarshal, 1477
  - tightMarshal1, 1477
  - tightMarshal2, 1478
  - tightUnmarshal, 1478
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1508
  - ~DataArrayResponseMarshaller, 1509
  - createObject, 1509
  - DataArrayResponseMarshaller, 1509
  - getDataSetType, 1509
  - looseMarshal, 1509
  - looseUnmarshal, 1510
  - tightMarshal1, 1511
  - tightMarshal2, 1511
  - tightUnmarshal, 1511

activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1573  
     ~DataResponseMarshaller, 1574  
     createObject, 1574  
     DataResponseMarshaller, 1574  
     getDataStructureType, 1574  
     looseMarshal, 1575  
     looseUnmarshal, 1575  
     tightMarshal1, 1576  
     tightMarshal2, 1576  
     tightUnmarshal, 1576  
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1708  
     ~DestinationInfoMarshaller, 1709  
     createObject, 1709  
     DestinationInfoMarshaller, 1709  
     getDataStructureType, 1709  
     looseMarshal, 1709  
     looseUnmarshal, 1710  
     tightMarshal1, 1710  
     tightMarshal2, 1711  
     tightUnmarshal, 1711  
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1741  
     ~DiscoveryEventMarshaller, 1742  
     createObject, 1742  
     DiscoveryEventMarshaller, 1742  
     getDataStructureType, 1742  
     looseMarshal, 1742  
     looseUnmarshal, 1743  
     tightMarshal1, 1743  
     tightMarshal2, 1744  
     tightUnmarshal, 1744  
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1825  
     ~ExceptionResponseMarshaller, 1826  
     createObject, 1826  
     ExceptionResponseMarshaller, 1826  
     getDataStructureType, 1826  
     looseMarshal, 1826  
     looseUnmarshal, 1827  
     tightMarshal1, 1827  
     tightMarshal2, 1828  
     tightUnmarshal, 1828  
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1919  
     ~FlushCommandMarshaller, 1920  
     createObject, 1920  
     FlushCommandMarshaller, 1920  
     getDataStructureType, 1920  
     DataResponseMarshaller, 1921  
     looseUnmarshal, 1921  
     tightMarshal1, 1922  
     tightMarshal2, 1922  
     tightUnmarshal, 1922  
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2073  
     ~IntegerResponseMarshaller, 2074  
     createObject, 2074  
     getDataStructureType, 2074  
     IntegerResponseMarshaller, 2074  
     looseMarshal, 2074  
     looseUnmarshal, 2075  
     tightMarshal1, 2075  
     tightMarshal2, 2076  
     tightUnmarshal, 2076  
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2139  
     ~JournalQueueAckMarshaller, 2140  
     createObject, 2140  
     getDataStructureType, 2140  
     JournalQueueAckMarshaller, 2140  
     looseMarshal, 2141  
     looseUnmarshal, 2141  
     tightMarshal1, 2142  
     tightMarshal2, 2142  
     tightUnmarshal, 2142  
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2168  
     ~JournalTopicAckMarshaller, 2169  
     createObject, 2169  
     getDataStructureType, 2169  
     JournalTopicAckMarshaller, 2169  
     looseMarshal, 2169  
     looseUnmarshal, 2170  
     tightMarshal1, 2170  
     tightMarshal2, 2170  
     tightUnmarshal, 2171  
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2190  
     ~JournalTraceMarshaller, 2191  
     createObject, 2191  
     getDataStructureType, 2191  
     JournalTraceMarshaller, 2191  
     looseMarshal, 2192  
     looseUnmarshal, 2192  
     tightMarshal1, 2193  
     tightMarshal2, 2193  
     tightUnmarshal, 2193

- activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2543
  - 2221
  - ~JournalTransactionMarshaller, 2222
  - createObject, 2222
  - getDataStructureType, 2223
  - JournalTransactionMarshaller, 2222
  - looseMarshal, 2223
  - looseUnmarshal, 2223
  - tightMarshal1, 2224
  - tightMarshal2, 2224
  - tightUnmarshal, 2225
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2583
  - 2249
  - ~KeepAliveInfoMarshaller, 2250
  - createObject, 2250
  - getDataStructureType, 2250
  - KeepAliveInfoMarshaller, 2250
  - looseMarshal, 2250
  - looseUnmarshal, 2251
  - tightMarshal1, 2251
  - tightMarshal2, 2252
  - tightUnmarshal, 2252
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2612
  - 2283
  - ~LastPartialCommandMarshaller, 2284
  - createObject, 2284
  - getDataStructureType, 2284
  - LastPartialCommandMarshaller, 2284
  - looseMarshal, 2284
  - looseUnmarshal, 2285
  - tightMarshal1, 2285
  - tightMarshal2, 2286
  - tightUnmarshal, 2286
- activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2648
  - 2330
  - ~LocalTransactionIdMarshaller, 2331
  - createObject, 2331
  - getDataStructureType, 2332
  - LocalTransactionIdMarshaller, 2331
  - looseMarshal, 2332
  - looseUnmarshal, 2332
  - tightMarshal1, 2333
  - tightMarshal2, 2333
  - tightUnmarshal, 2334
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2543
  - 2450
  - ~MarshallerFactory, 2450
  - configure, 2450
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2582
  - 2542
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2612
  - 2582
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2649
  - 2670
- activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2671
  - 2672

- tightMarshal1, 2672
- tightMarshal2, 2673
- tightUnmarshal, 2674
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 3041
  - 2716
  - ~MessagePullMarshaller, 2717
  - createObject, 2717
  - getDataStructureType, 2717
  - looseMarshal, 2717
  - looseUnmarshal, 2718
  - MessagePullMarshaller, 2717
  - tightMarshal1, 2718
  - tightMarshal2, 2719
  - tightUnmarshal, 2719
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2769
  - 2769
  - ~NetworkBridgeFilterMarshaller, 2770
  - createObject, 2770
  - getDataStructureType, 2770
  - looseMarshal, 2771
  - looseUnmarshal, 2771
  - NetworkBridgeFilterMarshaller, 2770
  - tightMarshal1, 2772
  - tightMarshal2, 2772
  - tightUnmarshal, 2772
- activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2891
  - 2891
  - ~PartialCommandMarshaller, 2892
  - createObject, 2893
  - getDataStructureType, 2893
  - looseMarshal, 2893
  - looseUnmarshal, 2894
  - PartialCommandMarshaller, 2892
  - tightMarshal1, 2894
  - tightMarshal2, 2895
  - tightUnmarshal, 2895
- activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 3008
  - 3008
  - ~ProducerAckMarshaller, 3009
  - createObject, 3009
  - getDataStructureType, 3009
  - looseMarshal, 3009
  - looseUnmarshal, 3010
  - ProducerAckMarshaller, 3009
  - tightMarshal1, 3010
  - tightMarshal2, 3011
  - tightUnmarshal, 3011
- activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3039
  - 3039
  - ~ProducerIdMarshaller, 3040
  - createObject, 3040
  - getDataStructureType, 3040
  - looseMarshal, 3040
  - MessagePullMarshaller, 3041
  - ProducerIdMarshaller, 3040
  - tightMarshal1, 3041
  - tightMarshal2, 3042
  - tightUnmarshal, 3042
- activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3056
  - 3056
  - ~ProducerInfoMarshaller, 3057
  - createObject, 3057
  - getDataStructureType, 3057
  - looseMarshal, 3057
  - NetworkBridgeFilterMarshaller, 3058
  - ProducerInfoMarshaller, 3057
  - tightMarshal1, 3058
  - tightMarshal2, 3059
  - tightUnmarshal, 3059
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 3153
  - 3153
  - ~RemoveInfoMarshaller, 3154
  - createObject, 3154
  - getDataStructureType, 3154
  - looseMarshal, 3155
  - looseUnmarshal, 3155
  - RemoveInfoMarshaller, 3154
  - tightMarshal1, 3156
  - tightMarshal2, 3156
  - tightUnmarshal, 3156
- activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3169
  - 3169
  - ~RemoveSubscriptionInfoMarshaller, 3170
  - createObject, 3170
  - getDataStructureType, 3171
  - looseMarshal, 3171
  - looseUnmarshal, 3171
  - RemoveSubscriptionInfoMarshaller, 3170
  - tightMarshal1, 3172
  - tightMarshal2, 3172
  - tightUnmarshal, 3173
- activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3201
  - 3201
  - ~ReplayCommandMarshaller, 3202
  - createObject, 3202
  - getDataStructureType, 3202
  - looseMarshal, 3202
  - looseUnmarshal, 3203
  - ReplayCommandMarshaller, 3202

- tightMarshal1, 3203
- tightMarshal2, 3204
- tightUnmarshal, 3204
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3255
  - ~ResponseMarshaller, 3256
  - createObject, 3256
  - getDataStructureType, 3256
  - looseMarshal, 3257
  - looseUnmarshal, 3257
  - ResponseMarshaller, 3256
  - tightMarshal1, 3258
  - tightMarshal2, 3258
  - tightUnmarshal, 3259
- activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3344
  - ~SessionIdMarshaller, 3345
  - createObject, 3345
  - getDataStructureType, 3345
  - looseMarshal, 3346
  - looseUnmarshal, 3346
  - SessionIdMarshaller, 3345
  - tightMarshal1, 3347
  - tightMarshal2, 3347
  - tightUnmarshal, 3347
- activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3360
  - ~SessionInfoMarshaller, 3361
  - createObject, 3361
  - getDataStructureType, 3361
  - looseMarshal, 3361
  - looseUnmarshal, 3362
  - SessionInfoMarshaller, 3361
  - tightMarshal1, 3362
  - tightMarshal2, 3363
  - tightUnmarshal, 3363
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 3424
  - ~ShutdownInfoMarshaller, 3425
  - createObject, 3425
  - getDataStructureType, 3425
  - looseMarshal, 3425
  - looseUnmarshal, 3426
  - ShutdownInfoMarshaller, 3425
  - tightMarshal1, 3426
  - tightMarshal2, 3427
  - tightUnmarshal, 3427
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 3624
  - ~SubscriptionInfoMarshaller, 3625
  - createObject, 3625
  - getDataStructureType, 3625
  - looseMarshal, 3626
  - SubscriptionInfoMarshaller, 3625
  - tightMarshal1, 3627
  - tightMarshal2, 3627
  - tightUnmarshal, 3627
- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3766
  - ~TransactionIdMarshaller, 3767
  - looseMarshal, 3767
  - looseUnmarshal, 3768
  - tightMarshal1, 3768
  - tightMarshal2, 3769
  - tightUnmarshal, 3769
  - TransactionIdMarshaller, 3767
- activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3793
  - ~TransactionInfoMarshaller, 3794
  - createObject, 3794
  - getDataStructureType, 3794
  - looseMarshal, 3795
  - looseUnmarshal, 3795
  - tightMarshal1, 3796
  - tightMarshal2, 3796
  - tightUnmarshal, 3796
  - TransactionInfoMarshaller, 3794
- activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3939
  - ~WireFormatInfoMarshaller, 3940
  - createObject, 3940
  - getDataStructureType, 3940
  - looseMarshal, 3940
  - looseUnmarshal, 3941
  - tightMarshal1, 3941
  - tightMarshal2, 3941
  - tightUnmarshal, 3942
  - WireFormatInfoMarshaller, 3940
- activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3976
  - ~XATransactionIdMarshaller, 3977
  - createObject, 3977
  - getDataStructureType, 3978
  - looseMarshal, 3978
  - looseUnmarshal, 3978
  - tightMarshal1, 3979
  - tightMarshal2, 3979
  - tightUnmarshal, 3980
  - XATransactionIdMarshaller, 3977

- activemq::wireformat::openwire::marshal::v2, 106
- activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 190
  - ~ActiveMQBlobMessageMarshaller, 191
  - ActiveMQBlobMessageMarshaller, 191
  - createObject, 191
  - getDataStructureType, 191
  - looseMarshal, 191
  - looseUnmarshal, 192
  - tightMarshal1, 192
  - tightMarshal2, 193
  - tightUnmarshal, 193
- activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 240
  - ~ActiveMQBytesMessageMarshaller, 241
  - ActiveMQBytesMessageMarshaller, 241
  - createObject, 241
  - getDataStructureType, 242
  - looseMarshal, 242
  - looseUnmarshal, 242
  - tightMarshal1, 243
  - tightMarshal2, 243
  - tightUnmarshal, 244
- activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 320
  - ~ActiveMQDestinationMarshaller, 321
  - ActiveMQDestinationMarshaller, 321
  - looseMarshal, 321
  - looseUnmarshal, 322
  - tightMarshal1, 322
  - tightMarshal2, 323
  - tightUnmarshal, 323
- activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 360
  - ~ActiveMQMapMessageMarshaller, 361
  - ActiveMQMapMessageMarshaller, 361
  - createObject, 361
  - getDataStructureType, 362
  - looseMarshal, 362
  - looseUnmarshal, 362
  - tightMarshal1, 363
  - tightMarshal2, 363
  - tightUnmarshal, 364
- activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 387
  - ~ActiveMQMessageMarshaller, 388
  - ActiveMQMessageMarshaller, 388
  - createObject, 388
  - getDataStructureType, 388
  - looseMarshal, 388
- ActiveMQBlobMessageMarshaller, 190
  - tightMarshal1, 389
  - tightMarshal2, 390
  - tightUnmarshal, 390
- activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 433
  - ~ActiveMQObjectMessageMarshaller, 434
  - ActiveMQObjectMessageMarshaller, 434
  - createObject, 434
  - getDataStructureType, 434
- ActiveMQBytesMessageMarshaller, 240
  - looseUnmarshal, 435
  - tightMarshal1, 435
  - tightMarshal2, 436
  - tightUnmarshal, 436
- activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 476
  - ~ActiveMQQueueMarshaller, 477
  - ActiveMQQueueMarshaller, 477
  - createObject, 477
  - getDataStructureType, 477
  - looseMarshal, 478
- ActiveMQDestinationMarshaller, 320
  - tightMarshal1, 479
  - tightMarshal2, 479
  - tightUnmarshal, 479
- activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 539
  - ~ActiveMQStreamMessageMarshaller, 540
  - ActiveMQStreamMessageMarshaller, 540
  - createObject, 540
  - getDataStructureType, 540
  - looseMarshal, 541
  - looseUnmarshal, 541
  - tightMarshal1, 542
  - tightMarshal2, 542
  - tightUnmarshal, 542
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 566
  - ~ActiveMQTempDestinationMarshaller, 567
  - ActiveMQTempDestinationMarshaller, 567
  - looseMarshal, 567
  - looseUnmarshal, 568

- tightMarshal1, 568
- tightMarshal2, 569
- tightUnmarshal, 569
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 594
  - ~ActiveMQTempQueueMarshaller, 595
  - ActiveMQTempQueueMarshaller, 595
  - createObject, 595
  - getDataStructureType, 596
  - looseMarshal, 596
  - looseUnmarshal, 596
  - tightMarshal1, 597
  - tightMarshal2, 597
  - tightUnmarshal, 598
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 623
  - ~ActiveMQTempTopicMarshaller, 624
  - ActiveMQTempTopicMarshaller, 624
  - createObject, 624
  - getDataStructureType, 624
  - looseMarshal, 624
  - looseUnmarshal, 625
  - tightMarshal1, 625
  - tightMarshal2, 626
  - tightUnmarshal, 626
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 656
  - ~ActiveMQTextMessageMarshaller, 657
  - ActiveMQTextMessageMarshaller, 657
  - createObject, 657
  - getDataStructureType, 657
  - looseMarshal, 657
  - looseUnmarshal, 658
  - tightMarshal1, 658
  - tightMarshal2, 659
  - tightUnmarshal, 659
- activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 684
  - ~ActiveMQTopicMarshaller, 685
  - ActiveMQTopicMarshaller, 685
  - createObject, 685
  - getDataStructureType, 685
  - looseMarshal, 685
  - looseUnmarshal, 686
  - tightMarshal1, 686
  - tightMarshal2, 687
  - tightUnmarshal, 687
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 764
  - ~BaseCommandMarshaller, 765
  - BaseCommandMarshaller, 765
  - looseMarshal, 765
  - looseUnmarshal, 766
  - tightMarshal1, 766
  - tightMarshal2, 768
  - tightUnmarshal, 769
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 852
  - ~BrokerIdMarshaller, 853
  - BrokerIdMarshaller, 853
  - createObject, 853
  - getDataStructureType, 853
  - looseMarshal, 853
  - looseUnmarshal, 854
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 883
  - ~BrokerInfoMarshaller, 884
  - BrokerInfoMarshaller, 884
  - createObject, 884
  - getDataStructureType, 884
  - looseMarshal, 884
  - looseUnmarshal, 885
- activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1262
  - ~ConnectionControlMarshaller, 1263
  - ConnectionControlMarshaller, 1263
  - createObject, 1263
  - getDataStructureType, 1263
  - looseMarshal, 1263
  - looseUnmarshal, 1264
- activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1270
  - ~ConnectionErrorMarshaller, 1271
  - ConnectionErrorMarshaller, 1271
  - createObject, 1271
  - getDataStructureType, 1271
  - looseMarshal, 1271
  - looseUnmarshal, 1272
- activemq::wireformat::openwire::marshal::v2::GetMetadataMarshaller, 1272
  - tightMarshal1, 1272
  - tightMarshal2, 1273
  - tightUnmarshal, 1273



- activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1462
  - ~ControlCommandMarshaller, 1463
  - ControlCommandMarshaller, 1463
  - createObject, 1464
  - getDataStructureType, 1464
  - looseMarshal, 1464
  - looseUnmarshal, 1464
  - tightMarshal1, 1465
  - tightMarshal2, 1465
  - tightUnmarshal, 1466
- activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1496
  - ~DataArrayResponseMarshaller, 1497
  - createObject, 1497
  - DataArrayResponseMarshaller, 1497
  - getDataStructureType, 1497
  - looseMarshal, 1497
  - looseUnmarshal, 1498
  - tightMarshal1, 1498
  - tightMarshal2, 1499
  - tightUnmarshal, 1499
- activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1561
  - ~DataResponseMarshaller, 1562
  - createObject, 1562
  - DataResponseMarshaller, 1562
  - getDataStructureType, 1562
  - looseMarshal, 1563
  - looseUnmarshal, 1563
  - tightMarshal1, 1563
  - tightMarshal2, 1564
  - tightUnmarshal, 1564
- activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1696
  - ~DestinationInfoMarshaller, 1697
  - createObject, 1697
  - DestinationInfoMarshaller, 1697
  - getDataStructureType, 1697
  - looseMarshal, 1697
  - looseUnmarshal, 1698
  - tightMarshal1, 1698
  - tightMarshal2, 1699
  - tightUnmarshal, 1699
- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1402
  - ~ConsumerIdMarshaller, 1403
  - ConsumerIdMarshaller, 1403
  - createObject, 1403
  - getDataStructureType, 1403
  - looseMarshal, 1403
  - looseUnmarshal, 1404
  - tightMarshal1, 1404
  - tightMarshal2, 1405
  - tightUnmarshal, 1405
- activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1434
  - ~ConsumerInfoMarshaller, 1435
  - ConsumerInfoMarshaller, 1435
  - createObject, 1435
  - getDataStructureType, 1436
- activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1301
  - ~ConnectionIdMarshaller, 1302
  - ConnectionIdMarshaller, 1302
  - createObject, 1302
  - getDataStructureType, 1302
  - looseMarshal, 1302
  - looseUnmarshal, 1303
  - tightMarshal1, 1303
  - tightMarshal2, 1303
  - tightUnmarshal, 1304
- activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1330
  - ~ConnectionInfoMarshaller, 1331
  - ConnectionInfoMarshaller, 1331
  - createObject, 1331
  - getDataStructureType, 1332
  - looseMarshal, 1332
  - looseUnmarshal, 1332
  - tightMarshal1, 1333
  - tightMarshal2, 1333
  - tightUnmarshal, 1334
- activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1373
  - ~ConsumerControlMarshaller, 1374
  - ConsumerControlMarshaller, 1374
  - createObject, 1375
  - getDataStructureType, 1375
  - looseMarshal, 1375
  - looseUnmarshal, 1375
  - tightMarshal1, 1376
  - tightMarshal2, 1376
  - tightUnmarshal, 1377
- activemq::wireformat::openwire::marshal::v2::ControlMarshaller, 1436
  - ~ControlMarshaller, 1437
  - ControlMarshaller, 1437
  - createObject, 1437
  - getDataStructureType, 1437
  - looseMarshal, 1437
  - looseUnmarshal, 1438
  - tightMarshal1, 1437
  - tightMarshal2, 1437
  - tightUnmarshal, 1438

activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1729  
     ~DiscoveryEventMarshaller, 1730  
     createObject, 1730  
     DiscoveryEventMarshaller, 1730  
     getDataStructureType, 1730  
     looseMarshal, 1730  
     looseUnmarshal, 1731  
     tightMarshal1, 1731  
     tightMarshal2, 1732  
     tightUnmarshal, 1732  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1809  
     ~ExceptionResponseMarshaller, 1810  
     createObject, 1810  
     ExceptionResponseMarshaller, 1810  
     getDataStructureType, 1810  
     looseMarshal, 1810  
     looseUnmarshal, 1811  
     tightMarshal1, 1811  
     tightMarshal2, 1812  
     tightUnmarshal, 1812  
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1907  
     ~FlushCommandMarshaller, 1908  
     createObject, 1908  
     FlushCommandMarshaller, 1908  
     getDataStructureType, 1908  
     looseMarshal, 1909  
     looseUnmarshal, 1909  
     tightMarshal1, 1910  
     tightMarshal2, 1910  
     tightUnmarshal, 1910  
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 2061  
     ~IntegerResponseMarshaller, 2062  
     createObject, 2062  
     getDataStructureType, 2062  
     IntegerResponseMarshaller, 2062  
     looseMarshal, 2062  
     looseUnmarshal, 2063  
     tightMarshal1, 2063  
     tightMarshal2, 2064  
     tightUnmarshal, 2064  
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2123  
     ~JournalQueueAckMarshaller, 2124  
     createObject, 2124  
     getDataStructureType, 2124  
     JournalQueueAckMarshaller, 2124  
     looseMarshal, 2125  
     looseUnmarshal, 2125  
     tightMarshal1, 2126  
     tightMarshal2, 2126  
     tightUnmarshal, 2126  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2152  
     ~JournalTopicAckMarshaller, 2153  
     createObject, 2153  
     getDataStructureType, 2153  
     JournalTopicAckMarshaller, 2153  
     looseMarshal, 2153  
     looseUnmarshal, 2154  
     tightMarshal1, 2154  
     tightMarshal2, 2154  
     tightUnmarshal, 2155  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2174  
     ~JournalTraceMarshaller, 2175  
     createObject, 2175  
     getDataStructureType, 2176  
     JournalTraceMarshaller, 2175  
     looseMarshal, 2176  
     looseUnmarshal, 2176  
     tightMarshal1, 2177  
     tightMarshal2, 2177  
     tightUnmarshal, 2177  
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2205  
     ~JournalTransactionMarshaller, 2206  
     createObject, 2206  
     getDataStructureType, 2207  
     JournalTransactionMarshaller, 2206  
     looseMarshal, 2207  
     looseUnmarshal, 2207  
     tightMarshal1, 2208  
     tightMarshal2, 2208  
     tightUnmarshal, 2209  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2233  
     ~KeepAliveInfoMarshaller, 2234  
     createObject, 2234  
     getDataStructureType, 2234  
     KeepAliveInfoMarshaller, 2234  
     looseMarshal, 2234  
     looseUnmarshal, 2235  
     tightMarshal1, 2235  
     tightMarshal2, 2236  
     tightUnmarshal, 2236

- activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2271
  - ~LastPartialCommandMarshaller, 2272
  - createObject, 2272
  - getDataStructureType, 2272
  - LastPartialCommandMarshaller, 2272
  - looseMarshal, 2272
  - looseUnmarshal, 2273
  - tightMarshal1, 2273
  - tightMarshal2, 2274
  - tightUnmarshal, 2274
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2314
  - ~LocalTransactionIdMarshaller, 2315
  - createObject, 2315
  - getDataStructureType, 2316
  - LocalTransactionIdMarshaller, 2315
  - looseMarshal, 2316
  - looseUnmarshal, 2316
  - tightMarshal1, 2317
  - tightMarshal2, 2317
  - tightUnmarshal, 2318
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2600
  - createObject, 2600
  - getDataStructureType, 2600
  - looseMarshal, 2601
  - looseUnmarshal, 2601
  - MessageDispatchNotificationMarshaller, 2600
  - tightMarshal1, 2601
  - tightMarshal2, 2602
  - tightUnmarshal, 2602
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2628
  - ~MessageIdMarshaller, 2629
  - createObject, 2629
  - getDataStructureType, 2629
  - looseMarshal, 2629
  - looseUnmarshal, 2630
  - MessageIdMarshaller, 2629
  - tightMarshal1, 2630
  - tightMarshal2, 2631
  - tightUnmarshal, 2631
- activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2450
  - ~MessageMarshaller, 2451
  - configure, 2451
- activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2530
  - ~MessageAckMarshaller, 2531
  - createObject, 2531
  - getDataStructureType, 2531
  - looseMarshal, 2532
  - looseUnmarshal, 2532
  - MessageAckMarshaller, 2531
  - tightMarshal1, 2533
  - tightMarshal2, 2533
  - tightUnmarshal, 2533
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2566
  - ~MessageDispatchMarshaller, 2567
  - createObject, 2567
  - getDataStructureType, 2568
  - looseMarshal, 2568
  - looseUnmarshal, 2568
  - MessageDispatchMarshaller, 2567
  - tightMarshal1, 2569
  - tightMarshal2, 2569
  - tightUnmarshal, 2570
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2599
  - createObject, 2600
  - getDataStructureType, 2600
  - looseMarshal, 2601
  - looseUnmarshal, 2601
  - MessageDispatchNotificationMarshaller, 2600
  - tightMarshal1, 2601
  - tightMarshal2, 2602
  - tightUnmarshal, 2602
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2700
  - ~MessagePullMarshaller, 2701
  - createObject, 2701
  - getDataStructureType, 2701
  - looseMarshal, 2701
  - MessagePullMarshaller, 2701
  - tightMarshal1, 2702
  - tightMarshal2, 2703
  - tightUnmarshal, 2703
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2749
  - ~NetworkBridgeFilterMarshaller, 2750
  - createObject, 2750
  - getDataStructureType, 2750
  - looseMarshal, 2751
  - NetworkBridgeFilterMarshaller, 2750

- tightMarshal1, 2752
- tightMarshal2, 2752
- tightUnmarshal, 2752
- activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2874
  - ~PartialCommandMarshaller, 2875
  - createObject, 2875
  - getDataStructureType, 2876
  - looseMarshal, 2876
  - looseUnmarshal, 2876
  - PartialCommandMarshaller, 2875
  - tightMarshal1, 2877
  - tightMarshal2, 2877
  - tightUnmarshal, 2878
- activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2988
  - ~ProducerAckMarshaller, 2989
  - createObject, 2989
  - getDataStructureType, 2989
  - looseMarshal, 2989
  - looseUnmarshal, 2990
  - ProducerAckMarshaller, 2989
  - tightMarshal1, 2990
  - tightMarshal2, 2991
  - tightUnmarshal, 2991
- activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3019
  - ~ProducerIdMarshaller, 3020
  - createObject, 3020
  - getDataStructureType, 3020
  - looseMarshal, 3020
  - looseUnmarshal, 3021
  - ProducerIdMarshaller, 3020
  - tightMarshal1, 3021
  - tightMarshal2, 3022
  - tightUnmarshal, 3022
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3052
  - ~ProducerInfoMarshaller, 3053
  - createObject, 3053
  - getDataStructureType, 3053
  - looseMarshal, 3053
  - looseUnmarshal, 3054
  - ProducerInfoMarshaller, 3053
  - tightMarshal1, 3054
  - tightMarshal2, 3055
  - tightUnmarshal, 3055
- activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3141
  - ~RemoveInfoMarshaller, 3142
  - createObject, 3142
  - getDataStructureType, 3142
  - looseMarshal, 3143
  - RemoveInfoMarshaller, 3142
  - tightMarshal1, 3144
  - tightMarshal2, 3144
  - tightUnmarshal, 3144
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3178
  - ~RemoveSubscriptionInfoMarshaller, 3179
  - createObject, 3179
  - getDataStructureType, 3179
  - looseMarshal, 3179
  - looseUnmarshal, 3180
  - RemoveSubscriptionInfoMarshaller, 3179
  - tightMarshal1, 3180
  - tightMarshal2, 3181
  - tightUnmarshal, 3181
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3205
  - ~ReplayCommandMarshaller, 3206
  - createObject, 3206
  - getDataStructureType, 3206
  - looseMarshal, 3206
  - looseUnmarshal, 3207
  - ReplayCommandMarshaller, 3206
  - tightMarshal1, 3207
  - tightMarshal2, 3208
  - tightUnmarshal, 3208
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3241
  - ~ResponseMarshaller, 3242
  - createObject, 3242
  - getDataStructureType, 3242
  - looseMarshal, 3243
  - looseUnmarshal, 3243
  - ResponseMarshaller, 3242
  - tightMarshal1, 3244
  - tightMarshal2, 3244
  - tightUnmarshal, 3245
- activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3324
  - ~SessionIdMarshaller, 3325
  - createObject, 3325
  - getDataStructureType, 3326
  - looseMarshal, 3326
  - looseUnmarshal, 3326
  - SessionIdMarshaller, 3325

- tightMarshal1, 3327
- tightMarshal2, 3327
- tightUnmarshal, 3327
- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3368
  - ~SessionInfoMarshaller, 3369
  - createObject, 3369
  - getDataStructureType, 3369
  - looseMarshal, 3369
  - looseUnmarshal, 3370
  - SessionInfoMarshaller, 3369
  - tightMarshal1, 3370
  - tightMarshal2, 3371
  - tightUnmarshal, 3371
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 3420
  - ~ShutdownInfoMarshaller, 3421
  - createObject, 3421
  - getDataStructureType, 3421
  - looseMarshal, 3421
  - looseUnmarshal, 3422
  - ShutdownInfoMarshaller, 3421
  - tightMarshal1, 3422
  - tightMarshal2, 3423
  - tightUnmarshal, 3423
- activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 3640
  - ~SubscriptionInfoMarshaller, 3641
  - createObject, 3641
  - getDataStructureType, 3641
  - looseMarshal, 3642
  - looseUnmarshal, 3642
  - SubscriptionInfoMarshaller, 3641
  - tightMarshal1, 3643
  - tightMarshal2, 3643
  - tightUnmarshal, 3643
- activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3770
  - ~TransactionIdMarshaller, 3771
  - looseMarshal, 3771
  - looseUnmarshal, 3772
  - tightMarshal1, 3772
  - tightMarshal2, 3773
  - tightUnmarshal, 3773
  - TransactionIdMarshaller, 3771
- activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3809
  - ~TransactionInfoMarshaller, 3810
  - createObject, 3810
  - getDataStructureType, 3810
- looseMarshal, 3811
- looseUnmarshal, 3811
- tightMarshal1, 3812
- tightMarshal2, 3812
- tightUnmarshal, 3812
- TransactionInfoMarshaller, 3810
- activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3931
  - ~WireFormatInfoMarshaller, 3932
  - createObject, 3932
  - getDataStructureType, 3932
  - looseMarshal, 3932
  - looseUnmarshal, 3933
  - tightMarshal1, 3933
  - tightMarshal2, 3934
  - tightUnmarshal, 3934
  - WireFormatInfoMarshaller, 3932
- activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3968
  - ~XATransactionIdMarshaller, 3969
  - createObject, 3969
  - getDataStructureType, 3970
  - looseMarshal, 3970
  - looseUnmarshal, 3970
  - tightMarshal1, 3971
  - tightMarshal2, 3972
  - tightUnmarshal, 3972
  - XATransactionIdMarshaller, 3969
- activemq::wireformat::openwire::marshal::v3, 110
- activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 177
  - ~ActiveMQBlobMessageMarshaller, 178
  - ActiveMQBlobMessageMarshaller, 178
  - createObject, 178
  - getDataStructureType, 179
  - looseMarshal, 179
  - looseUnmarshal, 179
  - tightMarshal1, 180
  - tightMarshal2, 180
  - tightUnmarshal, 181
- activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 220
  - ~ActiveMQBytesMessageMarshaller, 221
  - ActiveMQBytesMessageMarshaller, 221
  - createObject, 221
  - getDataStructureType, 222
  - looseMarshal, 222
  - looseUnmarshal, 222

- tightMarshal1, 223
- tightMarshal2, 223
- tightUnmarshal, 224
- activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 304
  - ~ActiveMQDestinationMarshaller, 305
  - ActiveMQDestinationMarshaller, 305
  - looseMarshal, 305
  - looseUnmarshal, 306
  - tightMarshal1, 306
  - tightMarshal2, 307
  - tightUnmarshal, 307
- activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 344
  - ~ActiveMQMapMessageMarshaller, 345
  - ActiveMQMapMessageMarshaller, 345
  - createObject, 345
  - getDataStructureType, 346
  - looseMarshal, 346
  - looseUnmarshal, 346
  - tightMarshal1, 347
  - tightMarshal2, 347
  - tightUnmarshal, 348
- activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 371
  - ~ActiveMQMessageMarshaller, 372
  - ActiveMQMessageMarshaller, 372
  - createObject, 372
  - getDataStructureType, 372
  - looseMarshal, 372
  - looseUnmarshal, 373
  - tightMarshal1, 373
  - tightMarshal2, 374
  - tightUnmarshal, 374
- activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 416
  - ~ActiveMQObjectMessageMarshaller, 417
  - ActiveMQObjectMessageMarshaller, 417
  - createObject, 417
  - getDataStructureType, 418
  - looseMarshal, 418
  - looseUnmarshal, 418
  - tightMarshal1, 419
  - tightMarshal2, 419
  - tightUnmarshal, 420
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 460
  - ~ActiveMQQueueMarshaller, 461
  - ActiveMQQueueMarshaller, 461
  - createObject, 461
  - getDataStructureType, 461
  - looseMarshal, 462
  - looseUnmarshal, 462
  - tightMarshal1, 463
  - tightMarshal2, 463
  - tightUnmarshal, 463
- activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 523
  - ~ActiveMQStreamMessageMarshaller, 524
  - ActiveMQStreamMessageMarshaller, 524
  - createObject, 524
  - getDataStructureType, 524
  - looseMarshal, 525
  - looseUnmarshal, 525
  - tightMarshal1, 525
  - tightMarshal2, 526
  - tightUnmarshal, 526
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 551
  - ~ActiveMQTempDestinationMarshaller, 552
  - ActiveMQTempDestinationMarshaller, 552
  - looseMarshal, 552
  - looseUnmarshal, 552
  - tightMarshal1, 553
  - tightMarshal2, 553
  - tightUnmarshal, 554
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 578
  - ~ActiveMQTempQueueMarshaller, 579
  - ActiveMQTempQueueMarshaller, 579
  - createObject, 579
  - getDataStructureType, 580
  - looseMarshal, 580
  - looseUnmarshal, 580
  - tightMarshal1, 581
  - tightMarshal2, 581
  - tightUnmarshal, 582
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 607
  - ~ActiveMQTempTopicMarshaller, 608
  - ActiveMQTempTopicMarshaller, 608
  - createObject, 608
  - getDataStructureType, 608
  - looseMarshal, 608
  - looseUnmarshal, 609

- tightMarshal1, 609
- tightMarshal2, 610
- tightUnmarshal, 610
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 635
  - ~ActiveMQTextMessageMarshaller, 636
  - ActiveMQTextMessageMarshaller, 636
  - createObject, 637
  - getDataStructureType, 637
  - looseMarshal, 637
  - looseUnmarshal, 637
  - tightMarshal1, 638
  - tightMarshal2, 638
  - tightUnmarshal, 639
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 664
  - ~ActiveMQTopicMarshaller, 665
  - ActiveMQTopicMarshaller, 665
  - createObject, 665
  - getDataStructureType, 665
  - looseMarshal, 665
  - looseUnmarshal, 666
  - tightMarshal1, 666
  - tightMarshal2, 667
  - tightUnmarshal, 667
- activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 730
  - ~BaseCommandMarshaller, 731
  - BaseCommandMarshaller, 731
  - looseMarshal, 731
  - looseUnmarshal, 732
  - tightMarshal1, 733
  - tightMarshal2, 734
  - tightUnmarshal, 735
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 832
  - ~BrokerIdMarshaller, 833
  - BrokerIdMarshaller, 833
  - createObject, 833
  - getDataStructureType, 833
  - looseMarshal, 833
  - looseUnmarshal, 834
  - tightMarshal1, 834
  - tightMarshal2, 835
  - tightUnmarshal, 835
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 862
  - ~BrokerInfoMarshaller, 863
  - BrokerInfoMarshaller, 863
  - createObject, 864
  - getDataStructureType, 864
  - looseMarshal, 864
  - looseUnmarshal, 864
  - tightMarshal1, 865
  - tightMarshal2, 865
  - tightUnmarshal, 866
- activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1242
  - ~ConnectionControlMarshaller, 1243
  - ConnectionControlMarshaller, 1243
  - createObject, 1243
  - getDataStructureType, 1243
  - looseMarshal, 1243
  - looseUnmarshal, 1244
- activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1274
  - ~ConnectionErrorMarshaller, 1275
  - ConnectionErrorMarshaller, 1275
  - createObject, 1275
  - getDataStructureType, 1275
  - looseMarshal, 1275
  - looseUnmarshal, 1276
- activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1305
  - ~ConnectionIdMarshaller, 1306
  - ConnectionIdMarshaller, 1306
  - createObject, 1306
  - getDataStructureType, 1306
  - looseUnmarshal, 1307
  - tightMarshal1, 1307
  - tightMarshal2, 1307
  - tightUnmarshal, 1308
- activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1335
  - ~ConnectionInfoMarshaller, 1336
  - ConnectionInfoMarshaller, 1336
  - createObject, 1336
  - getDataStructureType, 1336
  - looseUnmarshal, 1337
  - tightMarshal1, 1337
  - tightMarshal2, 1338
  - tightUnmarshal, 1338

activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1378  
     ~ConsumerControlMarshaller, 1379  
     ConsumerControlMarshaller, 1379  
     createObject, 1379  
     getDataStructureType, 1379  
     looseMarshal, 1379  
     looseUnmarshal, 1380  
     tightMarshal1, 1380  
     tightMarshal2, 1381  
     tightUnmarshal, 1381  
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1406  
     ~ConsumerIdMarshaller, 1407  
     ConsumerIdMarshaller, 1407  
     createObject, 1407  
     getDataStructureType, 1407  
     looseMarshal, 1407  
     looseUnmarshal, 1408  
     tightMarshal1, 1408  
     tightMarshal2, 1409  
     tightUnmarshal, 1409  
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1439  
     ~ConsumerInfoMarshaller, 1440  
     ConsumerInfoMarshaller, 1440  
     createObject, 1440  
     getDataStructureType, 1440  
     looseMarshal, 1440  
     looseUnmarshal, 1441  
     tightMarshal1, 1441  
     tightMarshal2, 1442  
     tightUnmarshal, 1442  
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1467  
     ~ControlCommandMarshaller, 1468  
     ControlCommandMarshaller, 1468  
     createObject, 1468  
     getDataStructureType, 1468  
     looseMarshal, 1468  
     looseUnmarshal, 1469  
     tightMarshal1, 1469  
     tightMarshal2, 1470  
     tightUnmarshal, 1470  
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1500  
     ~DataArrayResponseMarshaller, 1501  
     createObject, 1501  
     DataArrayResponseMarshaller, 1501  
     getDataStructureType, 1501  
     looseMarshal, 1501  
     looseUnmarshal, 1502  
     tightMarshal1, 1502  
     tightMarshal2, 1503  
     tightUnmarshal, 1503  
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1565  
     ~DataResponseMarshaller, 1566  
     createObject, 1566  
     DataResponseMarshaller, 1566  
     getDataStructureType, 1566  
     looseMarshal, 1567  
     looseUnmarshal, 1567  
     tightMarshal1, 1568  
     tightMarshal2, 1568  
     tightUnmarshal, 1568  
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1700  
     ~DestinationInfoMarshaller, 1701  
     createObject, 1701  
     DestinationInfoMarshaller, 1701  
     getDataStructureType, 1701  
     looseMarshal, 1701  
     looseUnmarshal, 1702  
     tightMarshal1, 1702  
     tightMarshal2, 1703  
     tightUnmarshal, 1703  
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1733  
     ~DiscoveryEventMarshaller, 1734  
     createObject, 1734  
     DiscoveryEventMarshaller, 1734  
     getDataStructureType, 1734  
     looseMarshal, 1734  
     looseUnmarshal, 1735  
     tightMarshal1, 1735  
     tightMarshal2, 1736  
     tightUnmarshal, 1736  
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1813  
     ~ExceptionResponseMarshaller, 1814  
     createObject, 1814  
     ExceptionResponseMarshaller, 1814  
     getDataStructureType, 1814  
     looseMarshal, 1814  
     looseUnmarshal, 1815  
     tightMarshal1, 1815  
     tightMarshal2, 1816  
     tightUnmarshal, 1816



- activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1911
  - ~FlushCommandMarshaller, 1912
  - createObject, 1912
  - FlushCommandMarshaller, 1912
  - getDataStructureType, 1912
  - looseMarshal, 1913
  - looseUnmarshal, 1913
  - tightMarshal1, 1914
  - tightMarshal2, 1914
  - tightUnmarshal, 1914
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2065
  - ~IntegerResponseMarshaller, 2066
  - createObject, 2066
  - getDataStructureType, 2066
  - IntegerResponseMarshaller, 2066
  - looseMarshal, 2066
  - looseUnmarshal, 2067
  - tightMarshal1, 2067
  - tightMarshal2, 2068
  - tightUnmarshal, 2068
- activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2131
  - ~JournalQueueAckMarshaller, 2132
  - createObject, 2132
  - getDataStructureType, 2132
  - JournalQueueAckMarshaller, 2132
  - looseMarshal, 2133
  - looseUnmarshal, 2133
  - tightMarshal1, 2134
  - tightMarshal2, 2134
  - tightUnmarshal, 2134
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2156
  - ~JournalTopicAckMarshaller, 2157
  - createObject, 2157
  - getDataStructureType, 2157
  - JournalTopicAckMarshaller, 2157
  - looseMarshal, 2157
  - looseUnmarshal, 2158
  - tightMarshal1, 2158
  - tightMarshal2, 2158
  - tightUnmarshal, 2159
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2178
  - ~JournalTraceMarshaller, 2179
  - createObject, 2179
  - getDataStructureType, 2179
  - JournalTraceMarshaller, 2179
- activemq::wireformat::openwire::marshal::v3::LocalCommandMarshaller, 2180
  - looseUnmarshal, 2180
  - tightMarshal1, 2181
  - tightMarshal2, 2181
  - tightUnmarshal, 2181
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2209
  - ~JournalTransactionMarshaller, 2210
  - createObject, 2210
  - getDataStructureType, 2211
  - JournalTransactionMarshaller, 2210
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 2211
  - looseMarshal, 2211
  - looseUnmarshal, 2211
  - tightMarshal1, 2212
  - tightMarshal2, 2212
  - tightUnmarshal, 2213
- activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2237
  - ~KeepAliveInfoMarshaller, 2238
  - createObject, 2238
  - getDataStructureType, 2238
  - KeepAliveInfoMarshaller, 2238
  - looseMarshal, 2238
  - looseUnmarshal, 2239
  - tightMarshal1, 2239
  - tightMarshal2, 2240
  - tightUnmarshal, 2240
- activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2267
  - ~LastPartialCommandMarshaller, 2268
  - createObject, 2268
  - getDataStructureType, 2268
  - LastPartialCommandMarshaller, 2268
  - looseMarshal, 2268
  - looseUnmarshal, 2269
  - tightMarshal1, 2269
  - tightMarshal2, 2270
  - tightUnmarshal, 2270
- activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2318
  - ~LocalTransactionIdMarshaller, 2319
  - createObject, 2319
  - getDataStructureType, 2320
  - LocalTransactionIdMarshaller, 2319
  - looseMarshal, 2320
  - looseUnmarshal, 2320
  - tightMarshal1, 2321
  - tightMarshal2, 2321
  - tightUnmarshal, 2322

activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2447  
     ~MarshallerFactory, 2448  
     configure, 2448  
 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2534  
     ~MessageAckMarshaller, 2535  
     createObject, 2535  
     getDataStructureType, 2535  
     looseMarshal, 2536  
     looseUnmarshal, 2536  
     MessageAckMarshaller, 2535  
     tightMarshal1, 2537  
     tightMarshal2, 2537  
     tightUnmarshal, 2537  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2570  
     ~MessageDispatchMarshaller, 2571  
     createObject, 2571  
     getDataStructureType, 2572  
     looseMarshal, 2572  
     looseUnmarshal, 2572  
     MessageDispatchMarshaller, 2571  
     tightMarshal1, 2573  
     tightMarshal2, 2573  
     tightUnmarshal, 2574  
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2603  
     ~MessageDispatchNotificationMarshaller, 2604  
     createObject, 2604  
     getDataStructureType, 2605  
     looseMarshal, 2605  
     looseUnmarshal, 2605  
     MessageDispatchNotificationMarshaller, 2604  
     tightMarshal1, 2606  
     tightMarshal2, 2606  
     tightUnmarshal, 2607  
 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2640  
     ~MessageIdMarshaller, 2641  
     createObject, 2641  
     getDataStructureType, 2641  
     looseMarshal, 2641  
     looseUnmarshal, 2642  
     MessageIdMarshaller, 2641  
     tightMarshal1, 2642  
     tightMarshal2, 2643  
     tightUnmarshal, 2643  
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2657  
     ~MessageMarshaller, 2658  
     looseMarshal, 2658  
     looseUnmarshal, 2659  
     MessageMarshaller, 2658  
     tightMarshal1, 2659  
     tightMarshal2, 2660  
     tightUnmarshal, 2661  
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2708  
     ~MessagePullMarshaller, 2709  
     createObject, 2709  
     getDataStructureType, 2709  
     looseMarshal, 2709  
     looseUnmarshal, 2710  
     MessagePullMarshaller, 2709  
     tightMarshal1, 2710  
     tightMarshal2, 2711  
     tightUnmarshal, 2711  
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2761  
     ~NetworkBridgeFilterMarshaller, 2762  
     createObject, 2762  
     getDataStructureType, 2762  
     looseMarshal, 2763  
     looseUnmarshal, 2763  
     NetworkBridgeFilterMarshaller, 2762  
     tightMarshal1, 2764  
     tightMarshal2, 2764  
     tightUnmarshal, 2764  
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2883  
     ~PartialCommandMarshaller, 2884  
     createObject, 2884  
     getDataStructureType, 2884  
     looseMarshal, 2884  
     looseUnmarshal, 2885  
     PartialCommandMarshaller, 2884  
     tightMarshal1, 2885  
     tightMarshal2, 2886  
     tightUnmarshal, 2886  
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2996  
     ~ProducerAckMarshaller, 2997  
     createObject, 2997  
     getDataStructureType, 2997  
     looseMarshal, 2997  
     looseUnmarshal, 2998  
     ProducerAckMarshaller, 2997

- tightMarshal1, 2998
- tightMarshal2, 2999
- tightUnmarshal, 2999
- activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3027
  - ~ProducerIdMarshaller, 3028
  - createObject, 3028
  - getDataStructureType, 3028
  - looseMarshal, 3028
  - looseUnmarshal, 3029
  - ProducerIdMarshaller, 3028
  - tightMarshal1, 3029
  - tightMarshal2, 3030
  - tightUnmarshal, 3030
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3064
  - ~ProducerInfoMarshaller, 3065
  - createObject, 3065
  - getDataStructureType, 3065
  - looseMarshal, 3065
  - looseUnmarshal, 3066
  - ProducerInfoMarshaller, 3065
  - tightMarshal1, 3066
  - tightMarshal2, 3067
  - tightUnmarshal, 3067
- activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3149
  - ~RemoveInfoMarshaller, 3150
  - createObject, 3150
  - getDataStructureType, 3150
  - looseMarshal, 3151
  - looseUnmarshal, 3151
  - RemoveInfoMarshaller, 3150
  - tightMarshal1, 3152
  - tightMarshal2, 3152
  - tightUnmarshal, 3152
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3174
  - ~RemoveSubscriptionInfoMarshaller, 3175
  - createObject, 3175
  - getDataStructureType, 3175
  - looseMarshal, 3175
  - looseUnmarshal, 3176
  - RemoveSubscriptionInfoMarshaller, 3175
  - tightMarshal1, 3176
  - tightMarshal2, 3177
  - tightUnmarshal, 3177
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3209
  - ~ReplayCommandMarshaller, 3210
  - createObject, 3210
  - getDataStructureType, 3210
  - looseMarshal, 3210
  - looseUnmarshal, 3211
  - ReplayCommandMarshaller, 3210
  - tightMarshal1, 3211
  - tightMarshal2, 3212
  - tightUnmarshal, 3212
- activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3250
  - ~ResponseMarshaller, 3251
  - createObject, 3251
  - getDataStructureType, 3252
  - looseMarshal, 3252
  - looseUnmarshal, 3253
  - ResponseMarshaller, 3251
  - tightMarshal1, 3253
  - tightMarshal2, 3254
  - tightUnmarshal, 3254
- activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3340
  - ~SessionIdMarshaller, 3341
  - createObject, 3341
  - getDataStructureType, 3341
  - looseMarshal, 3342
  - looseUnmarshal, 3342
  - SessionIdMarshaller, 3341
  - tightMarshal1, 3343
  - tightMarshal2, 3343
  - tightUnmarshal, 3343
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3364
  - ~SessionInfoMarshaller, 3365
  - createObject, 3365
  - getDataStructureType, 3365
  - looseMarshal, 3365
  - looseUnmarshal, 3366
  - SessionInfoMarshaller, 3365
  - tightMarshal1, 3366
  - tightMarshal2, 3367
  - tightUnmarshal, 3367
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 3432
  - ~ShutdownInfoMarshaller, 3433
  - createObject, 3433
  - getDataStructureType, 3433
  - looseMarshal, 3433
  - ShutdownInfoMarshaller, 3433

- tightMarshal1, 3434
- tightMarshal2, 3435
- tightUnmarshal, 3435
- activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 3620
  - ~SubscriptionInfoMarshaller, 3621
  - createObject, 3621
  - getDataStructureType, 3621
  - looseMarshal, 3622
  - looseUnmarshal, 3622
  - SubscriptionInfoMarshaller, 3621
  - tightMarshal1, 3623
  - tightMarshal2, 3623
  - tightUnmarshal, 3623
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3774
  - ~TransactionIdMarshaller, 3775
  - looseMarshal, 3775
  - looseUnmarshal, 3775
  - tightMarshal1, 3776
  - tightMarshal2, 3776
  - tightUnmarshal, 3777
  - TransactionIdMarshaller, 3775
- activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3797
  - ~TransactionInfoMarshaller, 3798
  - createObject, 3798
  - getDataStructureType, 3798
  - looseMarshal, 3799
  - looseUnmarshal, 3799
  - tightMarshal1, 3800
  - tightMarshal2, 3800
  - tightUnmarshal, 3800
  - TransactionInfoMarshaller, 3798
- activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3943
  - ~WireFormatInfoMarshaller, 3944
  - createObject, 3944
  - getDataStructureType, 3944
  - looseMarshal, 3944
  - looseUnmarshal, 3945
  - tightMarshal1, 3945
  - tightMarshal2, 3945
  - tightUnmarshal, 3946
  - WireFormatInfoMarshaller, 3944
- activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3980
  - ~XATransactionIdMarshaller, 3981
  - createObject, 3981
  - getDataStructureType, 3982
- looseMarshal, 3982
- looseUnmarshal, 3982
- tightMarshal1, 3983
- tightUnmarshal, 3984
- XATransactionIdMarshaller, 3981
- activemq::wireformat::openwire::marshal::v4, 113
- activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 186
  - ~ActiveMQBlobMessageMarshaller, 187
  - ActiveMQBlobMessageMarshaller, 187
  - createObject, 187
  - getDataStructureType, 187
- looseMarshal, 187
- looseUnmarshal, 188
- tightMarshal1, 188
- tightMarshal2, 189
- tightUnmarshal, 189
- activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 228
  - ~ActiveMQBytesMessageMarshaller, 229
- ActiveMQBytesMessageMarshaller, 229
- createObject, 229
- getDataStructureType, 230
- looseMarshal, 230
- looseUnmarshal, 230
- tightMarshal1, 231
- tightMarshal2, 231
- tightUnmarshal, 232
- activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMessageMarshaller, 312
  - ~ActiveMQDestinationMarshaller, 313
- ActiveMQDestinationMarshaller, 313
- looseMarshal, 313
- looseUnmarshal, 314
- tightMarshal1, 314
- tightMarshal2, 315
- tightUnmarshal, 315
- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 352
  - ~ActiveMQMapMessageMarshaller, 353
  - ActiveMQMapMessageMarshaller, 353
  - createObject, 353
  - getDataStructureType, 354
- looseMarshal, 354
- looseUnmarshal, 354
- tightMarshal1, 355
- tightMarshal2, 355

- tightUnmarshal, 356
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 379
  - ~ActiveMQMessageMarshaller, 380
  - ActiveMQMessageMarshaller, 380
  - createObject, 380
  - getDataStructureType, 380
  - looseMarshal, 380
  - looseUnmarshal, 381
  - tightMarshal1, 381
  - tightMarshal2, 382
  - tightUnmarshal, 382
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 425
  - ~ActiveMQObjectMessageMarshaller, 426
  - ActiveMQObjectMessageMarshaller, 426
  - createObject, 426
  - getDataStructureType, 426
  - looseMarshal, 426
  - looseUnmarshal, 427
  - tightMarshal1, 427
  - tightMarshal2, 428
  - tightUnmarshal, 428
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 468
  - ~ActiveMQQueueMarshaller, 469
  - ActiveMQQueueMarshaller, 469
  - createObject, 469
  - getDataStructureType, 469
  - looseMarshal, 470
  - looseUnmarshal, 470
  - tightMarshal1, 471
  - tightMarshal2, 471
  - tightUnmarshal, 471
- activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 531
  - ~ActiveMQStreamMessageMarshaller, 532
  - ActiveMQStreamMessageMarshaller, 532
  - createObject, 532
  - getDataStructureType, 532
  - looseMarshal, 533
  - looseUnmarshal, 533
  - tightMarshal1, 534
  - tightMarshal2, 534
  - tightUnmarshal, 534
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 558
  - ~ActiveMQTempDestinationMarshaller, 559
  - ActiveMQTempDestinationMarshaller, 559
  - looseMarshal, 560
  - looseUnmarshal, 560
  - tightMarshal1, 560
  - tightMarshal2, 561
  - tightUnmarshal, 562
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 586
  - ~ActiveMQTempQueueMarshaller, 587
  - ActiveMQTempQueueMarshaller, 587
  - createObject, 587
  - getDataStructureType, 588
  - looseMarshal, 588
  - looseUnmarshal, 588
  - tightMarshal1, 589
  - tightMarshal2, 589
  - tightUnmarshal, 590
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 611
  - ~ActiveMQTempTopicMarshaller, 612
  - ActiveMQTempTopicMarshaller, 612
  - createObject, 612
  - getDataStructureType, 612
  - looseMarshal, 612
  - looseUnmarshal, 613
  - tightMarshal1, 613
  - tightMarshal2, 614
  - tightUnmarshal, 614
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 640
  - ~ActiveMQTextMessageMarshaller, 641
  - ActiveMQTextMessageMarshaller, 641
  - createObject, 641
  - getDataStructureType, 641
  - looseMarshal, 641
  - looseUnmarshal, 642
  - tightMarshal1, 642
  - tightMarshal2, 643
  - tightUnmarshal, 643
- activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 668
  - ~ActiveMQTopicMarshaller, 669
  - ActiveMQTopicMarshaller, 669
  - createObject, 669
  - getDataStructureType, 669
  - looseMarshal, 669
  - looseUnmarshal, 670

- tightMarshal1, 670
- tightMarshal2, 671
- tightUnmarshal, 671
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 737
  - ~BaseCommandMarshaller, 738
  - BaseCommandMarshaller, 738
  - looseMarshal, 738
  - looseUnmarshal, 739
  - tightMarshal1, 740
  - tightMarshal2, 741
  - tightUnmarshal, 742
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 836
  - ~BrokerIdMarshaller, 837
  - BrokerIdMarshaller, 837
  - createObject, 837
  - getDataStructureType, 837
  - looseMarshal, 837
  - looseUnmarshal, 838
  - tightMarshal1, 838
  - tightMarshal2, 839
  - tightUnmarshal, 839
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 867
  - ~BrokerInfoMarshaller, 868
  - BrokerInfoMarshaller, 868
  - createObject, 868
  - getDataStructureType, 868
  - looseMarshal, 868
  - looseUnmarshal, 869
  - tightMarshal1, 869
  - tightMarshal2, 870
  - tightUnmarshal, 870
- activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1246
  - ~ConnectionControlMarshaller, 1247
  - ConnectionControlMarshaller, 1247
  - createObject, 1247
  - getDataStructureType, 1247
  - looseMarshal, 1247
  - looseUnmarshal, 1248
  - tightMarshal1, 1248
  - tightMarshal2, 1249
  - tightUnmarshal, 1249
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1278
  - ~ConnectionErrorMarshaller, 1279
  - ConnectionErrorMarshaller, 1279
  - createObject, 1279
  - getDataStructureType, 1279
  - looseMarshal, 1279
  - looseUnmarshal, 1280
  - tightMarshal1, 1280
  - tightMarshal2, 1281
  - tightUnmarshal, 1281
- activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1309
  - ~ConnectionIdMarshaller, 1310
  - ConnectionIdMarshaller, 1310
  - createObject, 1310
  - getDataStructureType, 1310
  - looseMarshal, 1310
  - looseUnmarshal, 1311
  - tightMarshal1, 1311
  - tightMarshal2, 1311
  - tightUnmarshal, 1312
- activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1339
  - ~ConnectionInfoMarshaller, 1340
  - ConnectionInfoMarshaller, 1340
  - createObject, 1340
  - getDataStructureType, 1340
  - looseMarshal, 1340
  - looseUnmarshal, 1341
  - tightMarshal1, 1341
  - tightMarshal2, 1342
  - tightUnmarshal, 1342
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1382
  - ~ConsumerControlMarshaller, 1383
  - ConsumerControlMarshaller, 1383
  - createObject, 1383
  - getDataStructureType, 1383
  - looseMarshal, 1383
  - looseUnmarshal, 1384
  - tightMarshal1, 1384
  - tightMarshal2, 1385
  - tightUnmarshal, 1385
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1410
  - ~ConsumerIdMarshaller, 1411
  - ConsumerIdMarshaller, 1411
  - createObject, 1411
  - getDataStructureType, 1411
  - looseMarshal, 1411
  - looseUnmarshal, 1412
  - tightMarshal1, 1412
  - tightMarshal2, 1413
  - tightUnmarshal, 1413

- activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1443
  - ~ConsumerInfoMarshaller, 1444
  - ConsumerInfoMarshaller, 1444
  - createObject, 1444
  - getDataStructureType, 1444
  - looseMarshal, 1444
  - looseUnmarshal, 1445
  - tightMarshal1, 1445
  - tightMarshal2, 1446
  - tightUnmarshal, 1446
- activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1471
  - ~ControlCommandMarshaller, 1472
  - ControlCommandMarshaller, 1472
  - createObject, 1472
  - getDataStructureType, 1472
  - looseMarshal, 1472
  - looseUnmarshal, 1473
  - tightMarshal1, 1473
  - tightMarshal2, 1474
  - tightUnmarshal, 1474
- activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1504
  - ~DataArrayResponseMarshaller, 1505
  - createObject, 1505
  - DataArrayResponseMarshaller, 1505
  - getDataStructureType, 1505
  - looseMarshal, 1506
  - looseUnmarshal, 1506
  - tightMarshal1, 1507
  - tightMarshal2, 1507
  - tightUnmarshal, 1507
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1569
  - ~DataResponseMarshaller, 1570
  - createObject, 1570
  - DataResponseMarshaller, 1570
  - getDataStructureType, 1570
  - looseMarshal, 1571
  - looseUnmarshal, 1571
  - tightMarshal1, 1572
  - tightMarshal2, 1572
  - tightUnmarshal, 1572
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1704
  - ~DestinationInfoMarshaller, 1705
  - createObject, 1705
  - DestinationInfoMarshaller, 1705
  - getDataStructureType, 1705
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1737
  - ~DiscoveryEventMarshaller, 1738
  - createObject, 1738
  - DiscoveryEventMarshaller, 1738
  - getDataStructureType, 1738
  - looseMarshal, 1738
  - looseUnmarshal, 1739
  - tightMarshal1, 1739
  - tightMarshal2, 1740
  - tightUnmarshal, 1740
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1821
  - ~ExceptionResponseMarshaller, 1822
  - createObject, 1822
  - ExceptionResponseMarshaller, 1822
  - getDataStructureType, 1822
  - looseMarshal, 1822
  - looseUnmarshal, 1823
  - tightMarshal1, 1823
  - tightMarshal2, 1824
  - tightUnmarshal, 1824
- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1915
  - ~FlushCommandMarshaller, 1916
  - createObject, 1916
  - FlushCommandMarshaller, 1916
  - getDataStructureType, 1916
  - looseMarshal, 1917
  - looseUnmarshal, 1917
  - tightMarshal1, 1918
  - tightMarshal2, 1918
  - tightUnmarshal, 1918
- activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 2069
  - ~IntegerResponseMarshaller, 2070
  - createObject, 2070
  - getDataStructureType, 2070
  - IntegerResponseMarshaller, 2070
  - looseMarshal, 2070
  - looseUnmarshal, 2071
  - tightMarshal1, 2071
  - tightMarshal2, 2072
  - tightUnmarshal, 2072
- activemq::wireformat::openwire::marshal::v4::LooseMarshal, 2055
  - looseMarshal, 2055
  - looseUnmarshal, 1706
  - tightMarshal1, 1706
  - tightMarshal2, 1707
  - tightUnmarshal, 1707

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen



- tightMarshal1, 2581
- tightMarshal2, 2581
- tightUnmarshal, 2582
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2607
  - ~MessageDispatchNotificationMarshaller, 2608
- createObject, 2608
- getDataStructureType, 2609
- looseMarshal, 2609
- looseUnmarshal, 2609
- MessageDispatchNotificationMarshaller, 2608
- tightMarshal1, 2610
- tightMarshal2, 2610
- tightUnmarshal, 2611
- activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2632
  - ~MessageIdMarshaller, 2633
- createObject, 2633
- getDataStructureType, 2633
- looseMarshal, 2633
- looseUnmarshal, 2634
- MessageIdMarshaller, 2633
- tightMarshal1, 2634
- tightMarshal2, 2635
- tightUnmarshal, 2635
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2666
  - ~MessageMarshaller, 2667
- looseMarshal, 2667
- looseUnmarshal, 2667
- MessageMarshaller, 2667
- tightMarshal1, 2668
- tightMarshal2, 2669
- tightUnmarshal, 2669
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2712
  - ~MessagePullMarshaller, 2713
- createObject, 2713
- getDataStructureType, 2713
- looseMarshal, 2713
- looseUnmarshal, 2714
- MessagePullMarshaller, 2713
- tightMarshal1, 2714
- tightMarshal2, 2715
- tightUnmarshal, 2715
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2765
  - ~NetworkBridgeFilterMarshaller, 2766
- createObject, 2766
- getDataStructureType, 2766
- looseMarshal, 2767
- MessageDispatchNotificationMarshaller, 2767
- NetworkBridgeFilterMarshaller, 2766
- tightMarshal1, 2768
- tightMarshal2, 2768
- tightUnmarshal, 2768
- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2887
  - ~PartialCommandMarshaller, 2888
- createObject, 2888
- getDataStructureType, 2889
- looseMarshal, 2889
- looseUnmarshal, 2889
- PartialCommandMarshaller, 2888
- tightMarshal1, 2890
- tightMarshal2, 2890
- tightUnmarshal, 2891
- activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2992
  - ~ProducerAckMarshaller, 2993
- createObject, 2993
- getDataStructureType, 2993
- looseMarshal, 2993
- looseUnmarshal, 2994
- ProducerAckMarshaller, 2993
- tightMarshal1, 2994
- tightMarshal2, 2995
- tightUnmarshal, 2995
- activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3023
  - ~ProducerIdMarshaller, 3024
- createObject, 3024
- getDataStructureType, 3024
- looseMarshal, 3024
- looseUnmarshal, 3025
- ProducerIdMarshaller, 3024
- tightMarshal1, 3025
- tightMarshal2, 3026
- tightUnmarshal, 3026
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3047
  - ~ProducerInfoMarshaller, 3048
- createObject, 3049
- getDataStructureType, 3049
- looseMarshal, 3049
- looseUnmarshal, 3049
- ProducerInfoMarshaller, 3048
- tightMarshal1, 3050

- tightMarshal2, 3050
- tightUnmarshal, 3051
- activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3160
  - 3161
  - ~RemoveInfoMarshaller, 3162
  - createObject, 3162
  - getDataStructureType, 3162
  - looseMarshal, 3163
  - looseUnmarshal, 3163
  - RemoveInfoMarshaller, 3162
  - tightMarshal1, 3164
  - tightMarshal2, 3164
  - tightUnmarshal, 3164
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3190
  - 3191
  - ~RemoveSubscriptionInfoMarshaller, 3191
  - createObject, 3191
  - getDataStructureType, 3191
  - looseMarshal, 3191
  - looseUnmarshal, 3192
  - RemoveSubscriptionInfoMarshaller, 3191
  - tightMarshal1, 3192
  - tightMarshal2, 3193
  - tightUnmarshal, 3193
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3197
  - 3198
  - ~ReplayCommandMarshaller, 3198
  - createObject, 3198
  - getDataStructureType, 3198
  - looseMarshal, 3198
  - looseUnmarshal, 3199
  - ReplayCommandMarshaller, 3198
  - tightMarshal1, 3199
  - tightMarshal2, 3200
  - tightUnmarshal, 3200
- activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3236
  - 3237
  - ~ResponseMarshaller, 3237
  - createObject, 3237
  - getDataStructureType, 3238
  - looseMarshal, 3238
  - looseUnmarshal, 3239
  - ResponseMarshaller, 3237
  - tightMarshal1, 3239
  - tightMarshal2, 3240
  - tightUnmarshal, 3240
- activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3328
  - 3329
  - ~SessionIdMarshaller, 3329
  - createObject, 3329
  - getDataStructureType, 3329
  - looseMarshal, 3330
  - looseUnmarshal, 3330
  - SessionIdMarshaller, 3329
  - tightMarshal1, 3331
  - tightMarshal2, 3331
  - tightUnmarshal, 3331
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3372
  - 3373
  - ~SessionInfoMarshaller, 3373
  - createObject, 3373
  - getDataStructureType, 3373
  - looseMarshal, 3373
  - looseUnmarshal, 3374
  - SessionInfoMarshaller, 3373
  - tightMarshal1, 3374
  - tightMarshal2, 3375
  - tightUnmarshal, 3375
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 3436
  - 3437
  - ~ShutdownInfoMarshaller, 3437
  - createObject, 3437
  - getDataStructureType, 3437
  - looseMarshal, 3437
  - looseUnmarshal, 3438
  - ShutdownInfoMarshaller, 3437
  - tightMarshal1, 3438
  - tightMarshal2, 3439
  - tightUnmarshal, 3439
- activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 3632
  - 3633
  - ~SubscriptionInfoMarshaller, 3633
  - createObject, 3633
  - getDataStructureType, 3633
  - looseMarshal, 3634
  - looseUnmarshal, 3634
  - SubscriptionInfoMarshaller, 3633
  - tightMarshal1, 3635
  - tightMarshal2, 3635
  - tightUnmarshal, 3635
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3778
  - 3779
  - ~TransactionIdMarshaller, 3779
  - looseMarshal, 3779
  - looseUnmarshal, 3779
  - tightMarshal1, 3780
  - tightMarshal2, 3780
  - tightUnmarshal, 3781
  - TransactionIdMarshaller, 3778

- activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3805
  - ~TransactionInfoMarshaller, 3806
  - createObject, 3806
  - getDataStructureType, 3806
  - looseMarshal, 3807
  - looseUnmarshal, 3807
  - tightMarshal1, 3808
  - tightMarshal2, 3808
  - tightUnmarshal, 3808
  - TransactionInfoMarshaller, 3806
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3935
  - ~WireFormatInfoMarshaller, 3936
  - createObject, 3936
  - getDataStructureType, 3936
  - looseMarshal, 3936
  - looseUnmarshal, 3937
  - tightMarshal1, 3937
  - tightMarshal2, 3937
  - tightUnmarshal, 3938
  - WireFormatInfoMarshaller, 3936
- activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3972
  - ~XATransactionIdMarshaller, 3973
  - createObject, 3973
  - getDataStructureType, 3974
  - looseMarshal, 3974
  - looseUnmarshal, 3974
  - tightMarshal1, 3975
  - tightMarshal2, 3975
  - tightUnmarshal, 3976
  - XATransactionIdMarshaller, 3973
- activemq::wireformat::openwire::marshal::v5, 116
- activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 194
  - ~ActiveMQBlobMessageMarshaller, 195
  - ActiveMQBlobMessageMarshaller, 195
  - createObject, 195
  - getDataStructureType, 195
  - looseMarshal, 195
  - looseUnmarshal, 196
  - tightMarshal1, 196
  - tightMarshal2, 197
  - tightUnmarshal, 197
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 232
  - ~ActiveMQBytesMessageMarshaller, 233
- ActiveMQBytesMessageMarshaller, 233
  - createObject, 233
  - getDataStructureType, 234
  - looseMarshal, 234
  - looseUnmarshal, 234
  - tightMarshal1, 235
  - tightMarshal2, 235
  - tightUnmarshal, 236
- activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 316
  - ~ActiveMQDestinationMarshaller, 317
- ActiveMQDestinationMarshaller, 317
  - looseMarshal, 317
  - looseUnmarshal, 318
  - tightMarshal1, 318
  - tightMarshal2, 319
  - tightUnmarshal, 319
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 356
  - ~ActiveMQMapMessageMarshaller, 357
  - ActiveMQMapMessageMarshaller, 357
  - createObject, 357
  - getDataStructureType, 358
  - looseMarshal, 358
  - looseUnmarshal, 358
  - tightMarshal1, 359
  - tightMarshal2, 359
  - tightUnmarshal, 360
- activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 383
  - ~ActiveMQMessageMarshaller, 384
  - ActiveMQMessageMarshaller, 384
  - createObject, 384
  - getDataStructureType, 384
  - looseMarshal, 384
- ActiveMQMessageMarshaller, 384
  - tightMarshal1, 385
  - tightMarshal2, 386
  - tightUnmarshal, 386
- activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 429
  - ~ActiveMQObjectMessageMarshaller, 430
  - ActiveMQObjectMessageMarshaller, 430
  - createObject, 430
  - getDataStructureType, 430
  - looseMarshal, 430
  - looseUnmarshal, 431
  - tightMarshal1, 431
  - tightMarshal2, 432

- tightUnmarshal, 432
- activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMarshaller, 472
  - ~ActiveMQQueueMarshaller, 473
  - ActiveMQQueueMarshaller, 473
  - createObject, 473
  - getDataStructureType, 473
  - looseMarshal, 474
  - looseUnmarshal, 474
  - tightMarshal1, 475
  - tightMarshal2, 475
  - tightUnmarshal, 475
- activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 535
  - ~ActiveMQStreamMessageMarshaller, 536
  - ActiveMQStreamMessageMarshaller, 536
  - createObject, 536
  - getDataStructureType, 536
  - looseMarshal, 537
  - looseUnmarshal, 537
  - tightMarshal1, 538
  - tightMarshal2, 538
  - tightUnmarshal, 538
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 562
  - ~ActiveMQTempDestinationMarshaller, 563
  - ActiveMQTempDestinationMarshaller, 563
  - looseMarshal, 563
  - looseUnmarshal, 564
  - tightMarshal1, 564
  - tightMarshal2, 565
  - tightUnmarshal, 565
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 590
  - ~ActiveMQTempQueueMarshaller, 591
  - ActiveMQTempQueueMarshaller, 591
  - createObject, 591
  - getDataStructureType, 592
  - looseMarshal, 592
  - looseUnmarshal, 592
  - tightMarshal1, 593
  - tightMarshal2, 593
  - tightUnmarshal, 594
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 619
  - ~ActiveMQTempTopicMarshaller, 620
  - ActiveMQTempTopicMarshaller, 620
  - createObject, 620
  - getDataStructureType, 620
  - looseMarshal, 620
  - looseUnmarshal, 621
  - tightMarshal1, 621
  - tightMarshal2, 622
  - tightUnmarshal, 622
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 648
  - ~ActiveMQTextMessageMarshaller, 649
  - ActiveMQTextMessageMarshaller, 649
  - createObject, 649
  - getDataStructureType, 649
  - looseMarshal, 649
  - looseUnmarshal, 650
  - tightMarshal1, 650
  - tightMarshal2, 651
  - tightUnmarshal, 651
- activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 676
  - ~ActiveMQTopicMarshaller, 677
  - ActiveMQTopicMarshaller, 677
  - createObject, 677
  - getDataStructureType, 677
  - looseMarshal, 678
  - looseUnmarshal, 678
  - tightMarshal1, 678
  - tightMarshal2, 679
  - tightUnmarshal, 679
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 750
  - ~BaseCommandMarshaller, 751
  - BaseCommandMarshaller, 751
  - looseMarshal, 751
  - looseUnmarshal, 752
  - tightMarshal1, 752
  - tightMarshal2, 755
  - tightUnmarshal, 756
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 844
  - ~BrokerIdMarshaller, 845
  - BrokerIdMarshaller, 845
  - createObject, 845
  - getDataStructureType, 845
  - looseMarshal, 845
  - looseUnmarshal, 846
  - tightMarshal1, 846
  - tightMarshal2, 847
  - tightUnmarshal, 847

- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 848
  - 875
  - ~BrokerInfoMarshaller, 876
  - BrokerInfoMarshaller, 876
  - createObject, 876
  - getDataStructureType, 876
  - looseMarshal, 876
  - looseUnmarshal, 877
  - tightMarshal1, 877
  - tightMarshal2, 878
  - tightUnmarshal, 878
- activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1254
  - ~ConnectionControlMarshaller, 1255
  - ConnectionControlMarshaller, 1255
  - createObject, 1255
  - getDataStructureType, 1255
  - looseMarshal, 1255
  - looseUnmarshal, 1256
  - tightMarshal1, 1256
  - tightMarshal2, 1257
  - tightUnmarshal, 1257
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1286
  - ~ConnectionErrorMarshaller, 1287
  - ConnectionErrorMarshaller, 1287
  - createObject, 1287
  - getDataStructureType, 1287
  - looseMarshal, 1287
  - looseUnmarshal, 1288
  - tightMarshal1, 1288
  - tightMarshal2, 1289
  - tightUnmarshal, 1289
- activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1317
  - ~ConnectionIdMarshaller, 1318
  - ConnectionIdMarshaller, 1318
  - createObject, 1318
  - getDataStructureType, 1318
  - looseMarshal, 1318
  - looseUnmarshal, 1319
  - tightMarshal1, 1319
  - tightMarshal2, 1319
  - tightUnmarshal, 1320
- activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1347
  - ~ConnectionInfoMarshaller, 1348
  - ConnectionInfoMarshaller, 1348
  - createObject, 1348
  - getDataStructureType, 1348
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1390
  - ~ConsumerControlMarshaller, 1391
  - ConsumerControlMarshaller, 1391
  - createObject, 1391
  - getDataStructureType, 1391
  - looseMarshal, 1391
  - looseUnmarshal, 1392
  - tightMarshal1, 1392
  - tightMarshal2, 1393
  - tightUnmarshal, 1393
- activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1418
  - ~ConsumerIdMarshaller, 1419
  - ConsumerIdMarshaller, 1419
  - createObject, 1419
  - getDataStructureType, 1419
  - looseMarshal, 1419
  - looseUnmarshal, 1420
  - tightMarshal1, 1420
  - tightMarshal2, 1421
  - tightUnmarshal, 1421
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1451
  - ~ConsumerInfoMarshaller, 1452
  - ConsumerInfoMarshaller, 1452
  - createObject, 1452
  - getDataStructureType, 1452
  - looseMarshal, 1452
  - looseUnmarshal, 1453
  - tightMarshal1, 1453
  - tightMarshal2, 1454
  - tightUnmarshal, 1454
- activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1479
  - ~ControlCommandMarshaller, 1480
  - ControlCommandMarshaller, 1480
  - createObject, 1480
  - getDataStructureType, 1480
  - looseMarshal, 1480
  - looseUnmarshal, 1481
  - tightMarshal1, 1481
  - tightMarshal2, 1482
  - tightUnmarshal, 1482

activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1512  
     ~DataArrayResponseMarshaller, 1513  
     createObject, 1513  
     DataArrayResponseMarshaller, 1513  
     getDataStructureType, 1513  
     looseMarshal, 1514  
     looseUnmarshal, 1514  
     tightMarshal1, 1515  
     tightMarshal2, 1515  
     tightUnmarshal, 1515  
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1553  
     ~DataResponseMarshaller, 1554  
     createObject, 1554  
     DataResponseMarshaller, 1554  
     getDataStructureType, 1554  
     looseMarshal, 1554  
     looseUnmarshal, 1555  
     tightMarshal1, 1555  
     tightMarshal2, 1556  
     tightUnmarshal, 1556  
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1716  
     ~DestinationInfoMarshaller, 1717  
     createObject, 1717  
     DestinationInfoMarshaller, 1717  
     getDataStructureType, 1717  
     looseMarshal, 1717  
     looseUnmarshal, 1718  
     tightMarshal1, 1718  
     tightMarshal2, 1719  
     tightUnmarshal, 1719  
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1745  
     ~DiscoveryEventMarshaller, 1746  
     createObject, 1746  
     DiscoveryEventMarshaller, 1746  
     getDataStructureType, 1746  
     looseMarshal, 1746  
     looseUnmarshal, 1747  
     tightMarshal1, 1747  
     tightMarshal2, 1748  
     tightUnmarshal, 1748  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1817  
     ~ExceptionResponseMarshaller, 1818  
     createObject, 1818  
     ExceptionResponseMarshaller, 1818  
     getDataStructureType, 1818  
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1923  
     ~FlushCommandMarshaller, 1924  
     createObject, 1924  
     FlushCommandMarshaller, 1924  
     getDataStructureType, 1924  
     looseMarshal, 1925  
     looseUnmarshal, 1925  
     tightMarshal1, 1926  
     tightMarshal2, 1926  
     tightUnmarshal, 1926  
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 2077  
     ~IntegerResponseMarshaller, 2078  
     createObject, 2078  
     getDataStructureType, 2078  
     IntegerResponseMarshaller, 2078  
     looseMarshal, 2078  
     looseUnmarshal, 2079  
     tightMarshal1, 2079  
     tightMarshal2, 2080  
     tightUnmarshal, 2080  
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2127  
     ~JournalQueueAckMarshaller, 2128  
     createObject, 2128  
     getDataStructureType, 2128  
     JournalQueueAckMarshaller, 2128  
     looseMarshal, 2129  
     looseUnmarshal, 2129  
     tightMarshal1, 2130  
     tightMarshal2, 2130  
     tightUnmarshal, 2130  
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2148  
     ~JournalTopicAckMarshaller, 2149  
     createObject, 2149  
     getDataStructureType, 2149  
     JournalTopicAckMarshaller, 2149  
     looseMarshal, 2149  
     looseUnmarshal, 2150  
     tightMarshal1, 2150  
     tightMarshal2, 2150  
     tightUnmarshal, 2151

- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2194
  - ~JournalTraceMarshaller, 2195
  - createObject, 2195
  - getDataStructureType, 2195
  - JournalTraceMarshaller, 2195
  - looseMarshal, 2196
  - looseUnmarshal, 2196
  - tightMarshal1, 2197
  - tightMarshal2, 2197
  - tightUnmarshal, 2197
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2213
  - ~JournalTransactionMarshaller, 2214
  - createObject, 2214
  - getDataStructureType, 2215
  - JournalTransactionMarshaller, 2214
  - looseMarshal, 2215
  - looseUnmarshal, 2215
  - tightMarshal1, 2216
  - tightMarshal2, 2216
  - tightUnmarshal, 2217
- activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2245
  - ~KeepAliveInfoMarshaller, 2246
  - createObject, 2246
  - getDataStructureType, 2246
  - KeepAliveInfoMarshaller, 2246
  - looseMarshal, 2246
  - looseUnmarshal, 2247
  - tightMarshal1, 2247
  - tightMarshal2, 2248
  - tightUnmarshal, 2248
- activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2275
  - ~LastPartialCommandMarshaller, 2276
  - createObject, 2276
  - getDataStructureType, 2276
  - LastPartialCommandMarshaller, 2276
  - looseMarshal, 2276
  - looseUnmarshal, 2277
  - tightMarshal1, 2277
  - tightMarshal2, 2278
  - tightUnmarshal, 2278
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 2322
  - ~LocalTransactionIdMarshaller, 2323
  - createObject, 2323
  - getDataStructureType, 2324
  - LocalTransactionIdMarshaller, 2323
- activemq::wireformat::openwire::marshal::v5::LooseMarshal, 2324
  - looseUnmarshal, 2324
  - tightMarshal1, 2325
  - tightMarshal2, 2325
  - tightUnmarshal, 2326
- activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 2449
  - ~MarshallerFactory, 2449
  - configure, 2449
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2546
  - ~MessageAckMarshaller, 2547
  - createObject, 2547
  - getDataStructureType, 2547
  - looseMarshal, 2548
  - looseUnmarshal, 2548
  - MessageAckMarshaller, 2547
  - tightMarshal1, 2549
  - tightMarshal2, 2549
  - tightUnmarshal, 2549
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2574
  - ~MessageDispatchMarshaller, 2575
  - createObject, 2575
  - getDataStructureType, 2576
  - looseMarshal, 2576
  - looseUnmarshal, 2576
  - MessageDispatchMarshaller, 2575
  - tightMarshal1, 2577
  - tightMarshal2, 2577
  - tightUnmarshal, 2578
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2616
  - ~MessageDispatchNotificationMarshaller, 2617
  - createObject, 2617
  - getDataStructureType, 2617
  - looseMarshal, 2617
  - looseUnmarshal, 2618
  - MessageDispatchNotificationMarshaller, 2617
  - tightMarshal1, 2618
  - tightMarshal2, 2619
  - tightUnmarshal, 2619
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2636
  - ~MessageIdMarshaller, 2637
  - createObject, 2637
  - getDataStructureType, 2637
  - looseMarshal, 2637

- looseUnmarshal, 2638
- MessageIdMarshaller, 2637
- tightMarshal1, 2638
- tightMarshal2, 2639
- tightUnmarshal, 2639
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2653
  - ~MessageMarshaller, 2654
  - looseMarshal, 2654
  - looseUnmarshal, 2654
  - MessageMarshaller, 2654
  - tightMarshal1, 2655
  - tightMarshal2, 2656
  - tightUnmarshal, 2656
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2704
  - ~MessagePullMarshaller, 2705
  - createObject, 2705
  - getDataStructureType, 2705
  - looseMarshal, 2705
  - looseUnmarshal, 2706
  - MessagePullMarshaller, 2705
  - tightMarshal1, 2706
  - tightMarshal2, 2707
  - tightUnmarshal, 2707
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2757
  - ~NetworkBridgeFilterMarshaller, 2758
  - createObject, 2758
  - getDataStructureType, 2758
  - looseMarshal, 2759
  - looseUnmarshal, 2759
  - NetworkBridgeFilterMarshaller, 2758
  - tightMarshal1, 2760
  - tightMarshal2, 2760
  - tightUnmarshal, 2760
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2878
  - ~PartialCommandMarshaller, 2879
  - createObject, 2880
  - getDataStructureType, 2880
  - looseMarshal, 2880
  - looseUnmarshal, 2881
  - PartialCommandMarshaller, 2879
  - tightMarshal1, 2881
  - tightMarshal2, 2882
  - tightUnmarshal, 2882
- activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 3000
  - ~ProducerAckMarshaller, 3001
  - createObject, 3001
  - getDataStructureType, 3001
  - looseMarshal, 3001
  - looseUnmarshal, 3002
  - ProducerAckMarshaller, 3001
  - tightMarshal1, 3002
  - tightMarshal2, 3003
  - tightUnmarshal, 3003
- activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3031
  - ~ProducerIdMarshaller, 3032
  - createObject, 3032
  - getDataStructureType, 3032
  - looseMarshal, 3032
  - looseUnmarshal, 3033
  - ProducerIdMarshaller, 3032
  - tightMarshal1, 3033
  - tightMarshal2, 3034
  - tightUnmarshal, 3034
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3060
  - ~ProducerInfoMarshaller, 3061
  - createObject, 3061
  - getDataStructureType, 3061
  - looseMarshal, 3061
  - looseUnmarshal, 3062
  - ProducerInfoMarshaller, 3061
  - tightMarshal1, 3062
  - tightMarshal2, 3063
  - tightUnmarshal, 3063
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3157
  - ~RemoveInfoMarshaller, 3158
  - createObject, 3158
  - getDataStructureType, 3158
  - looseMarshal, 3159
  - looseUnmarshal, 3159
  - RemoveInfoMarshaller, 3158
  - tightMarshal1, 3160
  - tightMarshal2, 3160
  - tightUnmarshal, 3160
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3186
  - ~RemoveSubscriptionInfoMarshaller, 3187
  - createObject, 3187
  - getDataStructureType, 3187
  - looseMarshal, 3187
  - looseUnmarshal, 3188
  - RemoveSubscriptionInfoMarshaller, 3187



Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 366
  - 3984
  - ~XATransactionIdMarshaller, 3985
  - createObject, 3985
  - getDataStructureType, 3986
  - looseMarshal, 3986
  - looseUnmarshal, 3986
  - tightMarshal1, 3987
  - tightMarshal2, 3987
  - tightUnmarshal, 3988
  - XATransactionIdMarshaller, 3985
- activemq::wireformat::openwire::marshal::v6, 119
- activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 198
  - ~ActiveMQBlobMessageMarshaller, 199
  - ActiveMQBlobMessageMarshaller, 199
  - createObject, 199
  - getDataStructureType, 199
  - looseMarshal, 199
  - looseUnmarshal, 200
  - tightMarshal1, 200
  - tightMarshal2, 201
  - tightUnmarshal, 201
- activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 236
  - ~ActiveMQBytesMessageMarshaller, 237
  - ActiveMQBytesMessageMarshaller, 237
  - createObject, 237
  - getDataStructureType, 238
  - looseMarshal, 238
  - looseUnmarshal, 238
  - tightMarshal1, 239
  - tightMarshal2, 239
  - tightUnmarshal, 240
- activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 324
  - ~ActiveMQDestinationMarshaller, 325
  - ActiveMQDestinationMarshaller, 325
  - looseMarshal, 325
  - looseUnmarshal, 326
  - tightMarshal1, 326
  - tightMarshal2, 327
  - tightUnmarshal, 327
- activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 364
  - ~ActiveMQMapMessageMarshaller, 365
  - ActiveMQMapMessageMarshaller, 365
  - createObject, 365
- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 366
  - looseMarshal, 366
  - looseUnmarshal, 366
  - tightMarshal1, 367
  - tightMarshal2, 367
  - tightUnmarshal, 368
- activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 391
  - ~ActiveMQMessageMarshaller, 392
  - ActiveMQMessageMarshaller, 392
  - createObject, 392
  - getDataStructureType, 392
  - looseMarshal, 392
- activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 437
  - ~ActiveMQObjectMessageMarshaller, 438
  - ActiveMQObjectMessageMarshaller, 438
  - createObject, 438
  - getDataStructureType, 438
  - looseMarshal, 438
  - looseUnmarshal, 439
  - tightMarshal1, 439
  - tightMarshal2, 440
  - tightUnmarshal, 440
- activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller, 480
  - ~ActiveMQQueueMarshaller, 481
  - ActiveMQQueueMarshaller, 481
  - createObject, 481
  - getDataStructureType, 481
  - looseMarshal, 482
  - looseUnmarshal, 482
  - tightMarshal1, 483
  - tightMarshal2, 483
  - tightUnmarshal, 483
- activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 543
  - ~ActiveMQStreamMessageMarshaller, 544
  - ActiveMQStreamMessageMarshaller, 544
  - createObject, 544
  - getDataStructureType, 544
  - looseMarshal, 545
  - looseUnmarshal, 545

- tightMarshal1, 546
- tightMarshal2, 546
- tightUnmarshal, 546
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller, 570
  - ~ActiveMQTempDestinationMarshaller, 571
- ActiveMQTempDestinationMarshaller, 571
- looseMarshal, 571
- looseUnmarshal, 572
- tightMarshal1, 572
- tightMarshal2, 573
- tightUnmarshal, 573
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 598
  - ~ActiveMQTempQueueMarshaller, 599
- ActiveMQTempQueueMarshaller, 599
- createObject, 599
- getDataStructureType, 600
- looseMarshal, 600
- looseUnmarshal, 600
- tightMarshal1, 601
- tightMarshal2, 601
- tightUnmarshal, 602
- activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 627
  - ~ActiveMQTempTopicMarshaller, 628
- ActiveMQTempTopicMarshaller, 628
- createObject, 628
- getDataStructureType, 628
- looseMarshal, 628
- looseUnmarshal, 629
- tightMarshal1, 629
- tightMarshal2, 630
- tightUnmarshal, 630
- activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 652
  - ~ActiveMQTextMessageMarshaller, 653
- ActiveMQTextMessageMarshaller, 653
- createObject, 653
- getDataStructureType, 653
- looseMarshal, 653
- looseUnmarshal, 654
- tightMarshal1, 654
- tightMarshal2, 655
- tightUnmarshal, 655
- activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller, 680
  - ~ActiveMQTopicMarshaller, 681
- ActiveMQTopicMarshaller, 681
- createObject, 681
- getDataStructureType, 681
- looseMarshal, 682
- tightMarshal1, 682
- tightMarshal2, 683
- tightUnmarshal, 683
- activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 757
  - ~BaseCommandMarshaller, 758
- BaseCommandMarshaller, 758
- looseMarshal, 758
- looseUnmarshal, 759
- tightMarshal1, 760
- tightMarshal2, 761
- tightUnmarshal, 763
- activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller, 848
  - ~BrokerIdMarshaller, 849
- BrokerIdMarshaller, 849
- createObject, 849
- getDataStructureType, 849
- looseMarshal, 849
- looseUnmarshal, 850
- tightMarshal1, 850
- tightMarshal2, 851
- tightUnmarshal, 851
- activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller, 879
  - ~BrokerInfoMarshaller, 880
- BrokerInfoMarshaller, 880
- createObject, 880
- getDataStructureType, 880
- looseMarshal, 880
- looseUnmarshal, 881
- tightMarshal1, 881
- tightMarshal2, 882
- tightUnmarshal, 882
- activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1258
  - ~ConnectionControlMarshaller, 1259
- ConnectionControlMarshaller, 1259
- createObject, 1259
- getDataStructureType, 1259
- looseMarshal, 1259
- looseUnmarshal, 1260
- tightMarshal1, 1260
- tightMarshal2, 1261
- tightUnmarshal, 1261

activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1423  
     1290  
     ~ControlCommandMarshaller, 1291  
     ControlCommandMarshaller, 1291  
     createObject, 1291  
     getDataStructureType, 1291  
     looseMarshal, 1291  
     looseUnmarshal, 1292  
     tightMarshal1, 1292  
     tightMarshal2, 1293  
     tightUnmarshal, 1293  
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1321  
     ~ConnectionIdMarshaller, 1322  
     ConnectionIdMarshaller, 1322  
     createObject, 1322  
     getDataStructureType, 1322  
     looseMarshal, 1322  
     looseUnmarshal, 1323  
     tightMarshal1, 1323  
     tightMarshal2, 1323  
     tightUnmarshal, 1324  
 activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1351  
     ~ConnectionInfoMarshaller, 1352  
     ConnectionInfoMarshaller, 1352  
     createObject, 1352  
     getDataStructureType, 1352  
     looseMarshal, 1352  
     looseUnmarshal, 1353  
     tightMarshal1, 1353  
     tightMarshal2, 1354  
     tightUnmarshal, 1354  
 activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1394  
     ~ConsumerControlMarshaller, 1395  
     ConsumerControlMarshaller, 1395  
     createObject, 1395  
     getDataStructureType, 1395  
     looseMarshal, 1395  
     looseUnmarshal, 1396  
     tightMarshal1, 1396  
     tightMarshal2, 1397  
     tightUnmarshal, 1397  
 activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1422  
     ~ConsumerIdMarshaller, 1423  
     ConsumerIdMarshaller, 1423  
     createObject, 1423  
     getDataStructureType, 1423  
 activemq::wireformat::openwire::marshal::v6::ConsumerInfoMarshaller, 1455  
     ~ConsumerInfoMarshaller, 1456  
     ConsumerInfoMarshaller, 1456  
     createObject, 1456  
     getDataStructureType, 1456  
     looseMarshal, 1456  
     looseUnmarshal, 1457  
     tightMarshal1, 1457  
     tightMarshal2, 1458  
     tightUnmarshal, 1458  
 activemq::wireformat::openwire::marshal::v6::ControlCommandMarshaller, 1483  
     ~ControlCommandMarshaller, 1484  
     ControlCommandMarshaller, 1484  
     createObject, 1484  
     getDataStructureType, 1484  
     looseMarshal, 1484  
     looseUnmarshal, 1485  
     tightMarshal1, 1485  
     tightMarshal2, 1486  
     tightUnmarshal, 1486  
 activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller, 1516  
     ~DataArrayResponseMarshaller, 1517  
     createObject, 1517  
     DataArrayResponseMarshaller, 1517  
     getDataStructureType, 1517  
     looseMarshal, 1518  
     looseUnmarshal, 1518  
     tightMarshal1, 1519  
     tightMarshal2, 1519  
     tightUnmarshal, 1519  
 activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1557  
     ~DataResponseMarshaller, 1558  
     createObject, 1558  
     DataResponseMarshaller, 1558  
     getDataStructureType, 1558  
     looseMarshal, 1558  
     looseUnmarshal, 1559  
     tightMarshal1, 1559  
     tightMarshal2, 1560  
     tightUnmarshal, 1560

- activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1712
  - ~DestinationInfoMarshaller, 1713
  - createObject, 1713
  - DestinationInfoMarshaller, 1713
  - getDataStructureType, 1713
  - looseMarshal, 1713
  - looseUnmarshal, 1714
  - tightMarshal1, 1714
  - tightMarshal2, 1715
  - tightUnmarshal, 1715
- activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1725
  - ~DiscoveryEventMarshaller, 1726
  - createObject, 1726
  - DiscoveryEventMarshaller, 1726
  - getDataStructureType, 1726
  - looseMarshal, 1726
  - looseUnmarshal, 1727
  - tightMarshal1, 1727
  - tightMarshal2, 1728
  - tightUnmarshal, 1728
- activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1804
  - ~ExceptionResponseMarshaller, 1805
  - createObject, 1805
  - ExceptionResponseMarshaller, 1805
  - getDataStructureType, 1806
  - looseMarshal, 1806
  - looseUnmarshal, 1806
  - tightMarshal1, 1807
  - tightMarshal2, 1807
  - tightUnmarshal, 1808
- activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 1903
  - ~FlushCommandMarshaller, 1904
  - createObject, 1904
  - FlushCommandMarshaller, 1904
  - getDataStructureType, 1904
  - looseMarshal, 1905
  - looseUnmarshal, 1905
  - tightMarshal1, 1906
  - tightMarshal2, 1906
  - tightUnmarshal, 1906
- activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 2057
  - ~IntegerResponseMarshaller, 2058
  - createObject, 2058
  - getDataStructureType, 2058
  - IntegerResponseMarshaller, 2058
- activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2119
  - ~JournalQueueAckMarshaller, 2120
  - createObject, 2120
  - getDataStructureType, 2121
  - JournalQueueAckMarshaller, 2120
- activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2160
  - ~JournalTopicAckMarshaller, 2161
  - createObject, 2161
  - getDataStructureType, 2161
  - JournalTopicAckMarshaller, 2161
- activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2182
  - ~JournalTraceMarshaller, 2183
  - createObject, 2183
  - getDataStructureType, 2183
  - JournalTraceMarshaller, 2183
- activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2201
  - ~JournalTransactionMarshaller, 2202
  - createObject, 2203
  - getDataStructureType, 2203
  - JournalTransactionMarshaller, 2202
- activemq::wireformat::openwire::marshal::v6::LooseMarshal, 2058
  - looseUnmarshal, 2059
  - tightMarshal1, 2059
  - tightMarshal2, 2060
  - tightUnmarshal, 2060
- activemq::wireformat::openwire::marshal::v6::LooseUnmarshal, 2059
  - tightMarshal1, 2059
  - tightMarshal2, 2060
  - tightUnmarshal, 2060
- activemq::wireformat::openwire::marshal::v6::LooseUnmarshal, 2121
  - tightMarshal1, 2122
  - tightMarshal2, 2122
  - tightUnmarshal, 2122
- activemq::wireformat::openwire::marshal::v6::LooseUnmarshal, 2162
  - tightMarshal1, 2162
  - tightMarshal2, 2162
  - tightUnmarshal, 2163
- activemq::wireformat::openwire::marshal::v6::LooseUnmarshal, 2184
  - tightMarshal1, 2185
  - tightMarshal2, 2185
  - tightUnmarshal, 2185
- activemq::wireformat::openwire::marshal::v6::LooseUnmarshal, 2203
  - tightMarshal1, 2204
  - tightMarshal2, 2204
  - tightUnmarshal, 2205

activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2587  
     2228  
     ~KeepAliveInfoMarshaller, 2229  
     createObject, 2229  
     getDataStructureType, 2230  
     KeepAliveInfoMarshaller, 2229  
     looseMarshal, 2230  
     looseUnmarshal, 2230  
     tightMarshal1, 2231  
     tightMarshal2, 2231  
     tightUnmarshal, 2232  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2262  
     ~LastPartialCommandMarshaller, 2263  
     createObject, 2263  
     getDataStructureType, 2264  
     LastPartialCommandMarshaller, 2263  
     looseMarshal, 2264  
     looseUnmarshal, 2264  
     tightMarshal1, 2265  
     tightMarshal2, 2265  
     tightUnmarshal, 2266  
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2310  
     ~LocalTransactionIdMarshaller, 2311  
     createObject, 2311  
     getDataStructureType, 2312  
     LocalTransactionIdMarshaller, 2311  
     looseMarshal, 2312  
     looseUnmarshal, 2312  
     tightMarshal1, 2313  
     tightMarshal2, 2313  
     tightUnmarshal, 2314  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2447  
     ~MarshallerFactory, 2447  
     configure, 2447  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2526  
     ~MessageAckMarshaller, 2527  
     createObject, 2527  
     getDataStructureType, 2527  
     looseMarshal, 2528  
     looseUnmarshal, 2528  
     MessageAckMarshaller, 2527  
     tightMarshal1, 2528  
     tightMarshal2, 2529  
     tightUnmarshal, 2529  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2586  
     ~MessageDispatchMarshaller, 2587  
     createObject, 2587  
     getDataStructureType, 2588  
     looseMarshal, 2588  
     looseUnmarshal, 2588  
     MessageDispatchMarshaller, 2587  
     tightMarshal1, 2589  
     tightMarshal2, 2589  
     tightUnmarshal, 2590  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2595  
     ~MessageDispatchNotificationMarshaller, 2596  
     createObject, 2596  
     getDataStructureType, 2596  
     looseMarshal, 2596  
     looseUnmarshal, 2597  
     MessageDispatchNotificationMarshaller, 2596  
     tightMarshal1, 2597  
     tightMarshal2, 2598  
     tightUnmarshal, 2598  
 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2644  
     ~MessageIdMarshaller, 2645  
     createObject, 2645  
     getDataStructureType, 2645  
     looseMarshal, 2645  
     looseUnmarshal, 2646  
     MessageIdMarshaller, 2645  
     tightMarshal1, 2646  
     tightMarshal2, 2647  
     tightUnmarshal, 2647  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2674  
     ~MessageMarshaller, 2675  
     looseMarshal, 2675  
     looseUnmarshal, 2676  
     MessageMarshaller, 2675  
     tightMarshal1, 2677  
     tightMarshal2, 2677  
     tightUnmarshal, 2678  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2720  
     ~MessagePullMarshaller, 2721  
     createObject, 2721  
     getDataStructureType, 2721  
     looseMarshal, 2721  
     looseUnmarshal, 2722  
     MessagePullMarshaller, 2721

- tightMarshal1, 2722
- tightMarshal2, 2723
- tightUnmarshal, 2723
- activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2753
  - ~NetworkBridgeFilterMarshaller, 2754
  - createObject, 2754
  - getDataStructureType, 2754
  - looseMarshal, 2755
  - looseUnmarshal, 2755
  - NetworkBridgeFilterMarshaller, 2754
  - tightMarshal1, 2756
  - tightMarshal2, 2756
  - tightUnmarshal, 2756
- activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2870
  - ~PartialCommandMarshaller, 2871
  - createObject, 2871
  - getDataStructureType, 2871
  - looseMarshal, 2871
  - looseUnmarshal, 2872
  - PartialCommandMarshaller, 2871
  - tightMarshal1, 2872
  - tightMarshal2, 2873
  - tightUnmarshal, 2873
- activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3004
  - ~ProducerAckMarshaller, 3005
  - createObject, 3005
  - getDataStructureType, 3005
  - looseMarshal, 3005
  - looseUnmarshal, 3006
  - ProducerAckMarshaller, 3005
  - tightMarshal1, 3006
  - tightMarshal2, 3007
  - tightUnmarshal, 3007
- activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3035
  - ~ProducerIdMarshaller, 3036
  - createObject, 3036
  - getDataStructureType, 3036
  - looseMarshal, 3036
  - looseUnmarshal, 3037
  - ProducerIdMarshaller, 3036
  - tightMarshal1, 3037
  - tightMarshal2, 3038
  - tightUnmarshal, 3038
- activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3068
  - ~ProducerInfoMarshaller, 3069
  - createObject, 3069
  - getDataStructureType, 3069
  - looseMarshal, 3069
  - NetworkBridgeFilterMarshaller, 3070
  - ProducerInfoMarshaller, 3069
  - tightMarshal1, 3070
  - tightMarshal2, 3071
  - tightUnmarshal, 3071
- activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3145
  - ~RemoveInfoMarshaller, 3146
  - createObject, 3146
  - getDataStructureType, 3146
  - looseMarshal, 3147
  - looseUnmarshal, 3147
  - RemoveInfoMarshaller, 3146
  - tightMarshal1, 3148
  - tightMarshal2, 3148
  - tightUnmarshal, 3148
- activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3182
  - ~RemoveSubscriptionInfoMarshaller, 3183
  - createObject, 3183
  - getDataStructureType, 3183
  - looseMarshal, 3183
  - looseUnmarshal, 3184
  - RemoveSubscriptionInfoMarshaller, 3183
  - tightMarshal1, 3184
  - tightMarshal2, 3185
  - tightUnmarshal, 3185
- activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3213
  - ~ReplayCommandMarshaller, 3214
  - createObject, 3214
  - getDataStructureType, 3214
  - looseMarshal, 3214
  - looseUnmarshal, 3215
  - ReplayCommandMarshaller, 3214
  - tightMarshal1, 3215
  - tightMarshal2, 3216
  - tightUnmarshal, 3216
- activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3260
  - ~ResponseMarshaller, 3261
  - createObject, 3261
  - getDataStructureType, 3261
  - looseMarshal, 3261
  - looseUnmarshal, 3262
  - ResponseMarshaller, 3261

- tightMarshal1, 3262
- tightMarshal2, 3263
- tightUnmarshal, 3263
- activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3332
  - ~SessionIdMarshaller, 3333
  - createObject, 3333
  - getDataStructureType, 3333
  - looseMarshal, 3334
  - looseUnmarshal, 3334
  - SessionIdMarshaller, 3333
  - tightMarshal1, 3335
  - tightMarshal2, 3335
  - tightUnmarshal, 3335
- activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3352
  - ~SessionInfoMarshaller, 3353
  - createObject, 3353
  - getDataStructureType, 3353
  - looseMarshal, 3353
  - looseUnmarshal, 3354
  - SessionInfoMarshaller, 3353
  - tightMarshal1, 3354
  - tightMarshal2, 3355
  - tightUnmarshal, 3355
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3416
  - ~ShutdownInfoMarshaller, 3417
  - createObject, 3417
  - getDataStructureType, 3417
  - looseMarshal, 3417
  - looseUnmarshal, 3418
  - ShutdownInfoMarshaller, 3417
  - tightMarshal1, 3418
  - tightMarshal2, 3419
  - tightUnmarshal, 3419
- activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3636
  - ~SubscriptionInfoMarshaller, 3637
  - createObject, 3637
  - getDataStructureType, 3637
  - looseMarshal, 3638
  - looseUnmarshal, 3638
  - SubscriptionInfoMarshaller, 3637
  - tightMarshal1, 3639
  - tightMarshal2, 3639
  - tightUnmarshal, 3639
- activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3781
  - ~TransactionIdMarshaller, 3782
- looseMarshal, 3782
- looseUnmarshal, 3783
- tightMarshal1, 3783
- tightUnmarshal, 3784
- TransactionIdMarshaller, 3782
- activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3801
  - ~TransactionInfoMarshaller, 3802
  - createObject, 3802
  - getDataStructureType, 3802
  - looseMarshal, 3803
  - looseUnmarshal, 3803
  - tightMarshal1, 3804
  - tightUnmarshal, 3804
  - TransactionInfoMarshaller, 3802
- activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3927
  - ~WireFormatInfoMarshaller, 3928
  - createObject, 3928
  - getDataStructureType, 3928
  - looseMarshal, 3928
  - looseUnmarshal, 3929
  - tightMarshal1, 3929
  - tightUnmarshal, 3929
  - WireFormatInfoMarshaller, 3928
- activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 3964
  - ~XATransactionIdMarshaller, 3965
  - createObject, 3965
  - getDataStructureType, 3966
  - looseMarshal, 3966
  - looseUnmarshal, 3966
  - tightMarshal1, 3967
  - tightUnmarshal, 3968
  - XATransactionIdMarshaller, 3965
- activemq::wireformat::openwire::OpenWireFormat, 2837
  - ~OpenWireFormat, 2840
  - addMarshaller, 2840
  - createNegotiator, 2840
  - DEFAULT\_VERSION, 2849
  - destroyMarshalers, 2840
  - doUnmarshal, 2840
  - getCouldBeMarshalled, 2841
  - getMaxInactivityDuration, 2841
  - getMaxInactivityDurationInitialDelay, 2841



- getPreferredWireFormatInfo, 2841
- getVersion, 2842
- hasNegotiator, 2842
- inReceive, 2842
- isCacheEnabled, 2842
- isSizePrefixDisabled, 2843
- isStackTraceEnabled, 2843
- isTcpNoDelayEnabled, 2843
- isTightEncodingEnabled, 2843
- looseMarshalNestedObject, 2843
- looseUnmarshalNestedObject, 2844
- marshal, 2844
- NULL\_TYPE, 2849
- OpenWireFormat, 2839
- renegotiateWireFormat, 2845
- setCacheEnabled, 2845
- setCacheSize, 2845
- setMaxInactivityDuration, 2845
- setMaxInactivityDurationInitialDelay, 2846
- setPreferredWireFormatInfo, 2846
- setSizePrefixDisabled, 2846
- setStackTraceEnabled, 2846
- setTcpNoDelayEnabled, 2847
- setTightEncodingEnabled, 2847
- setVersion, 2847
- tightMarshalNestedObject1, 2847
- tightMarshalNestedObject2, 2848
- tightUnmarshalNestedObject, 2848
- unmarshal, 2848
- activemq::wireformat::openwire::OpenWireFormat, 2849
  - ~OpenWireFormatFactory, 2850
  - createWireFormat, 2850
  - OpenWireFormatFactory, 2850
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2851
  - ~OpenWireFormatNegotiator, 2852
  - close, 2852
  - onCommand, 2852
  - oneway, 2852
  - onTransportException, 2853
  - OpenWireFormatNegotiator, 2851
  - request, 2853
  - start, 2854
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2854
  - ~OpenWireResponseBuilder, 2855
  - buildIncomingCommands, 2855
  - buildResponse, 2855
  - OpenWireResponseBuilder, 2855
- activemq::wireformat::openwire::utils, 122
- activemq::wireformat::openwire::utils::BooleanStream, 818
  - ~BooleanStream, 819
  - BooleanStream, 819
  - clear, 819
  - marshal, 819
  - marshalledSize, 819
  - readBoolean, 820
  - unmarshal, 820
  - writeBoolean, 820
- activemq::wireformat::openwire::utils::HexTable, 1947
  - ~HexTable, 1947
  - HexTable, 1947
  - operator[], 1947, 1948
  - size, 1948
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2689
  - ~MessagePropertyInterceptor, 2691
  - getBooleanProperty, 2691
  - getByteProperty, 2691
  - getDoubleProperty, 2691
  - getFloatProperty, 2692
  - getIntProperty, 2692
  - getLongProperty, 2692
  - getShortProperty, 2693
  - getStringProperty, 2693
  - MessagePropertyInterceptor, 2691
  - setBooleanProperty, 2693
  - setByteProperty, 2693
  - setDoubleProperty, 2694
  - setFloatProperty, 2694
  - setIntProperty, 2694
  - setLongProperty, 2694
  - setShortProperty, 2694
  - setStringProperty, 2695
- activemq::wireformat::stomp, 122
- activemq::wireformat::stomp::StompCommandConstants, 3571
  - ABORT, 3573
  - ACK, 3573
  - ACK\_AUTO, 3573
  - ACK\_CLIENT, 3573
  - ACK\_INDIVIDUAL, 3573
  - BEGIN, 3573
  - BYTES, 3573
  - COMMIT, 3573
  - CONNECT, 3573
  - CONNECTED, 3573

DISCONNECT, 3573  
 ERROR\_CMD, 3573  
 HEADER\_ACK, 3573  
 HEADER\_CLIENT\_ID, 3573  
 HEADER\_CONSUMERPRIORITY, 3573  
 HEADER\_CONTENTLENGTH, 3573  
 HEADER\_CORRELATIONID, 3574  
 HEADER\_DESTINATION, 3574  
 HEADER\_DISPATCH\_ASYNC, 3574  
 HEADER\_EXCLUSIVE, 3574  
 HEADER\_EXPIRES, 3574  
 HEADER\_ID, 3574  
 HEADER\_JMSPRIORITY, 3574  
 HEADER\_LOGIN, 3574  
 HEADER\_MAXPENDINGMSGLIMIT, 3574  
 HEADER\_MESSAGE, 3574  
 HEADER\_MESSAGEID, 3574  
 HEADER\_NOLOCAL, 3574  
 HEADER\_OLDSUBSCRIPTIONNAME, 3574  
 HEADER\_PASSWORD, 3574  
 HEADER\_PERSISTENT, 3574  
 HEADER\_PREFETCHSIZE, 3574  
 HEADER\_RECEIPT\_REQUIRED, 3575  
 HEADER\_RECEIPTID, 3575  
 HEADER\_REDELIVERED, 3575  
 HEADER\_REDELIVERYCOUNT, 3575  
 HEADER\_REPLYTO, 3575  
 HEADER\_REQUESTID, 3575  
 HEADER\_RESPONSEID, 3575  
 HEADER\_RETROACTIVE, 3575  
 HEADER\_SELECTOR, 3575  
 HEADER\_SESSIONID, 3575  
 HEADER\_SUBSCRIPTION, 3575  
 HEADER\_SUBSCRIPTIONNAME, 3575  
 HEADER\_TIMESTAMP, 3575  
 HEADER\_TRANSACTIONID, 3575  
 HEADER\_TRANSFORMATION, 3575  
 HEADER\_TRANSFORMATION\_ERROR, 3575  
 HEADER\_TYPE, 3576  
 MESSAGE, 3576  
 QUEUE\_PREFIX, 3576  
 RECEIPT, 3576  
 SEND, 3576  
 SUBSCRIBE, 3576  
 TEMPQUEUE\_PREFIX, 3576  
 TEMPTOPIC\_PREFIX, 3576  
 TEXT, 3576  
 TOPIC\_PREFIX, 3576  
 UNSUBSCRIBE, 3576  
 activemq::wireformat::stomp::StompFrame, 3576  
 ~StompFrame, 3578  
 clone, 3578  
 copy, 3578  
 fromStream, 3578  
 getBody, 3578, 3579  
 getBodyLength, 3579  
 getCommand, 3579  
 getProperties, 3579  
 getProperty, 3579  
 hasProperty, 3580  
 removeProperty, 3580  
 setBody, 3580  
 setCommand, 3580  
 setProperty, 3580  
 StompFrame, 3578  
 toStream, 3581  
 activemq::wireformat::stomp::StompHelper, 3581  
 ~StompHelper, 3582  
 convertConsumerId, 3582, 3583  
 convertDestination, 3583  
 convertMessageId, 3584  
 convertProducerId, 3584  
 convertProperties, 3585  
 convertTransactionId, 3585  
 StompHelper, 3582  
 activemq::wireformat::stomp::StompWireFormat, 3586  
 ~StompWireFormat, 3587  
 createNegotiator, 3587  
 getVersion, 3587  
 hasNegotiator, 3587  
 inReceive, 3588  
 marshal, 3588  
 setVersion, 3588  
 StompWireFormat, 3587  
 unmarshal, 3588  
 activemq::wireformat::stomp::StompWireFormatFactory, 3589  
 ~StompWireFormatFactory, 3590  
 createWireFormat, 3590  
 StompWireFormatFactory, 3590  
 activemq::wireformat::WireFormat, 3907  
 ~WireFormat, 3908  
 createNegotiator, 3908  
 getVersion, 3909

- hasNegotiator, 3909
- inReceive, 3909
- marshal, 3909
- setVersion, 3910
- unmarshal, 3910
- activemq::wireformat::WireFormatFactory, 3911
  - ~WireFormatFactory, 3911
  - createWireFormat, 3912
- activemq::wireformat::WireFormatNegotiator, 3946
  - ~WireFormatNegotiator, 3947
  - WireFormatNegotiator, 3947
- activemq::wireformat::WireFormatRegistry, 3947
  - ~WireFormatRegistry, 3948
  - findFactory, 3948
  - getInstance, 3949
  - getWireFormatNames, 3949
  - registerFactory, 3949
  - unregisterFactory, 3950
- ActiveMQBlobMessage
  - activemq::commands::ActiveMQBlobMessage, 174
- ActiveMQBlobMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 183
  - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 191
  - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 178
  - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 187
  - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 195
  - activemq::wireformat::openwire::marshal::v6::ActiveMQBlobMessageMarshaller, 199
- ActiveMQBytesMessage
  - activemq::commands::ActiveMQBytesMessage, 204
- ActiveMQBytesMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 225
  - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 241
  - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 221
  - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 229
  - activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 233
- activemq::wireformat::openwire::marshal::v6::ActiveMQBytesMessageMarshaller, 237
- ActiveMQConnection
  - activemq::core::ActiveMQConnection, 249
- ActiveMQConnectionFactory
  - activemq::core::ActiveMQConnectionFactory, 266
- ActiveMQConnectionMetaData
  - activemq::core::ActiveMQConnectionMetaData, 276
- ActiveMQConsumer
  - activemq::core::ActiveMQConsumer, 284
- ActiveMQCPP
  - activemq::library::ActiveMQCPP, 292
- ActiveMQDestination
  - activemq::commands::ActiveMQDestination, 296
- ActiveMQDestinationMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 309
  - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 321
  - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 305
  - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 310
  - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 317
  - activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller, 325
- ActiveMQException
  - activemq::core::ActiveMQException, 329
- ActiveMQMapMessage
  - activemq::commands::ActiveMQMapMessage, 333
- ActiveMQMapMessageMarshaller
  - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 349
  - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 361
  - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 345
  - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 353
  - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 357
  - activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller, 365

ActiveMQMessage  
     activemq::commands::ActiveMQMessage, 369  
 ActiveMQMessageMarshaller  
     activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 376  
     activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 388  
     activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 372  
     activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 380  
     activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 384  
     activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller, 392  
 ActiveMQMessageTemplate  
     activemq::commands::ActiveMQMessageTemplate, 398  
 ActiveMQObjectMessage  
     activemq::commands::ActiveMQObjectMessage, 414  
 ActiveMQObjectMessageMarshaller  
     activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 422  
     activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 434  
     activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 417  
     activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 426  
     activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 430  
     activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller, 438  
 ActiveMQProducer  
     activemq::core::ActiveMQProducer, 442  
 ActiveMQProperties  
     activemq::util::ActiveMQProperties, 450  
 ActiveMQQueue  
     activemq::commands::ActiveMQQueue, 454  
 ActiveMQQueueBrowser  
     activemq::core::ActiveMQQueueBrowser, 458  
 ActiveMQQueueMarshaller  
     activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 465  
     activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 477  
     activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 461  
     activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 469  
     activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 473  
     activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller, 481  
     activemq::core::ActiveMQSession, 488  
     activemq::core::ActiveMQSession, 503  
     activemq::core::ActiveMQSessionExecutor, 504  
     activemq::commands::ActiveMQStreamMessage, 509  
     activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 528  
     activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 540  
     activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 524  
     activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 532  
     activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 530  
     activemq::wireformat::openwire::marshal::v6::ActiveMQStreamMessageMarshaller, 544  
     ActiveMQTempDestination  
     activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestination, 548  
     activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestination, 556  
     activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestination, 567  
     activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestination, 552  
     activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestination, 559  
     activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestination, 563  
     activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestination, 571  
     activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestination, 571  
     activemq::commands::ActiveMQTempQueue, 570  
     ActiveMQTempQueueMarshaller

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 583  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 585  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 595  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 579  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 595  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 587  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 591  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 599  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller, 599  
 ActiveMQTempQueueMarshaller, 599  
 activemq::core::ActiveMQTransactionContext, 689  
 ActiveMQTempTopic, 689  
 activemq::commands::ActiveMQTempTopic, 603  
 ActiveMQTempTopicMarshaller, 1122  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 616  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 624  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 608  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 612  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 620  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller, 628  
 ActiveMQTextMessage, 150  
 activemq::commands::ActiveMQTextMessage, 165  
 ActiveMQTextMessageMarshaller, 1157  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 645  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 657  
 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 636  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 641  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 649  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller, 653  
 ActiveMQTopic, 488  
 activemq::commands::ActiveMQTopic, 3378  
 ActiveMQTopicMarshaller, 3378  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 673

- activemq::core::ActiveMQConnection, 249
- addHandler
  - decaf::util::logging::Logger, 2348
- additionalPredicate
  - activemq::commands::ConsumerInfo, 1433
- addLogger
  - decaf::util::logging::LogManager, 2366
- addMarshaller
  - activemq::wireformat::openwire::OpenWireFormat, 2840
- addMessageConsumer
  - activemq::cmsutil::ResourceLifecycleManager, 3226
- addMessageProducer
  - activemq::cmsutil::ResourceLifecycleManager, 3226
- addNetworkResource
  - decaf::internal::net::Network, 2745
- addProducer
  - activemq::core::ActiveMQConnection, 249
  - activemq::core::ActiveMQSession, 488
  - activemq::state::SessionState, 3378
- addProducerState
  - activemq::state::TransactionState, 3814
- addPropertyChangeListener
  - decaf::util::logging::LogManager, 2367
- addResource
  - decaf::internal::util::ResourceLifecycleManager, 3224
- address
  - decaf::net::SocketImpl, 3480
- addressBytes
  - decaf::net::InetAddress, 1981
- addSession
  - activemq::cmsutil::ResourceLifecycleManager, 3226
  - activemq::state::ConnectionState, 1359
- addSynchronization
  - activemq::core::ActiveMQTransactionContext, 689
- addTask
  - activemq::threads::CompositeTaskRunner, 1195
- addTempDestination
  - activemq::state::ConnectionState, 1359
- addTransactionState
  - activemq::state::ConnectionState, 1359
- addTransportListener
  - activemq::core::ActiveMQConnection, 250
- addURI
  - activemq::transport::CompositeTransport, 1197
  - activemq::transport::failover::FailoverTransport, 1838
  - activemq::transport::failover::URIPool, 3876
  - activemq::transport::failover::URIPool, 3876
- adler
  - activemq::stream\_s, 3991
- Adler32
  - decaf::util::zip::Adler32, 692
- advisory
  - activemq::commands::ActiveMQDestination, 302
- ADVISORY\_PREFIX
  - activemq::commands::ActiveMQDestination, 302
- after
  - decaf::util::Date, 1635
- afterCommit
  - activemq::core::Synchronization, 3659
- afterMarshal
  - activemq::commands::BaseDataStructure, 794
- afterMessageIsConsumed
  - activemq::core::ActiveMQConsumer, 285
- afterRollback
  - activemq::core::Synchronization, 3659
- afterUnmarshal
  - activemq::commands::BaseDataStructure, 794
  - activemq::commands::Message, 2480
  - activemq::commands::WireFormatInfo, 3914
- allocate
  - decaf::util::logging::Level, 2294
- decaf::nio::ByteBuffer, 1000
- decaf::nio::CharBuffer, 1092
- decaf::nio::DoubleBuffer, 1776

- decaf::nio::FloatBuffer, 1890
- decaf::nio::IntBuffer, 2029
- decaf::nio::LongBuffer, 2406
- decaf::nio::ShortBuffer, 3403
- AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION copy
  - CMSEExceptionSupport.h, 4085
- AMQ\_CATCH\_EXCEPTION\_CONVERT arrayOffset
  - activemq/exceptions/ExceptionDefines.h, 4052
- AMQ\_CATCH\_NOTHROW
  - activemq/exceptions/ExceptionDefines.h, 4052
- AMQ\_CATCH\_RETHROW
  - activemq/exceptions/ExceptionDefines.h, 4052
- AMQ\_CATCHALL\_NOTHROW
  - activemq/exceptions/ExceptionDefines.h, 4053
- AMQ\_CATCHALL\_THROW
  - activemq/exceptions/ExceptionDefines.h, 4053
- AMQCPP\_API
  - activemq/util/Config.h, 4086
- ANY\_CHILD ArrayPointer
  - activemq::commands::ActiveMQDestination::DestinationFilter, 1691
- ANY\_DESCENDENT
  - activemq::commands::ActiveMQDestination::DestinationFilter, 1691
- anyBytes
  - decaf::net::InetAddress, 1981
- append
  - decaf::io::Writer, 3952, 3953
  - decaf::lang::Appendable, 694, 695
  - decaf::nio::CharBuffer, 1093, 1094
- AprPool
  - decaf::internal::AprPool, 696
- array
  - decaf::internal::nio::ByteBuffer, 966
  - decaf::internal::nio::CharArrayBuffer, 1083
  - decaf::internal::nio::DoubleArrayBuffer, 1768
  - decaf::internal::nio::FloatArrayBuffer, 1882
  - decaf::internal::nio::IntArrayBuffer, 2021
  - decaf::internal::nio::LongArrayBuffer, 2398
  - decaf::internal::nio::ShortArrayBuffer, 3395
  - decaf::nio::ByteBuffer, 1001
  - decaf::nio::CharBuffer, 1094
  - decaf::nio::DoubleBuffer, 1776
  - decaf::nio::FloatBuffer, 1890
  - decaf::nio::IntBuffer, 2029
  - decaf::nio::LongBuffer, 2406
  - decaf::nio::ShortBuffer, 3404
  - decaf::lang::System, 3672, 3673
  - decaf::internal::nio::ByteBuffer, 966
  - decaf::internal::nio::CharArrayBuffer, 1083
  - decaf::internal::nio::DoubleArrayBuffer, 1768
  - decaf::internal::nio::FloatArrayBuffer, 1882
  - decaf::internal::nio::IntArrayBuffer, 2021
  - decaf::internal::nio::LongArrayBuffer, 2398
  - decaf::internal::nio::ShortArrayBuffer, 3396
  - decaf::nio::ByteBuffer, 1001
  - decaf::nio::CharBuffer, 1095
  - decaf::nio::DoubleBuffer, 1777
  - decaf::nio::FloatBuffer, 1891
  - decaf::nio::IntBuffer, 2029
  - decaf::nio::LongBuffer, 2406
  - decaf::nio::ShortBuffer, 3404
  - ArrayPointer, 699, 700
  - arrival
    - activemq::commands::Message, 2490
  - asCharBuffer
    - decaf::internal::nio::ByteBuffer, 967
  - asByteBuffer
    - decaf::nio::ByteBuffer, 1002
  - asciiToModifiedUtf8
    - activemq::util::MarshallingSupport, 2452
  - asDoubleBuffer
    - decaf::internal::nio::ByteBuffer, 967
    - decaf::nio::ByteBuffer, 1002
  - asFloatBuffer
    - decaf::internal::nio::ByteBuffer, 968
    - decaf::nio::ByteBuffer, 1002
  - asIntBuffer
    - decaf::internal::nio::ByteBuffer, 968
    - decaf::nio::ByteBuffer, 1003
  - asLongBuffer
    - decaf::internal::nio::ByteBuffer, 968
    - decaf::nio::ByteBuffer, 1003
  - asReadOnlyBuffer
    - decaf::internal::nio::ByteBuffer, 969
    - decaf::internal::nio::CharArrayBuffer, 1084
    - decaf::internal::nio::DoubleArrayBuffer, 1769
    - decaf::internal::nio::FloatArrayBuffer, 1883

- decaf::internal::nio::IntArrayBuffer, 2022
- decaf::internal::nio::LongArrayBuffer, 2399
- decaf::internal::nio::ShortArrayBuffer, 3396
- decaf::nio::ByteBuffer, 1003
- decaf::nio::CharBuffer, 1095
- decaf::nio::DoubleBuffer, 1777
- decaf::nio::FloatBuffer, 1891
- decaf::nio::IntBuffer, 2030
- decaf::nio::LongBuffer, 2407
- decaf::nio::ShortBuffer, 3405
- Assert
  - zutil.h, 4438
- asShortBuffer
  - decaf::internal::nio::ByteArrayBuffer, 969
  - decaf::nio::ByteBuffer, 1004
- AsyncSignalReadErrorTask
- activemq::transport::inactivity::InactivityMonitor, 1487–1489
- 1967
- AsyncWriteTask
  - activemq::transport::inactivity::InactivityMonitor, 1487–1489
  - 1967
- atEOF
  - decaf::util::zip::InflaterInputStream, 2001
- AtomicBoolean
  - decaf::util::concurrent::atomic::AtomicBoolean, 1834
  - 706
- AtomicInteger
  - decaf::util::concurrent::atomic::AtomicInteger, 1225
  - 709
- AtomicRefCounter
  - decaf::util::concurrent::atomic::AtomicRefCounter, 1226
  - 714
- AtomicReference
  - decaf::util::concurrent::atomic::AtomicReference, 716
- AUTO\_ACKNOWLEDGE
  - cms::Session, 3308
- avail\_in
  - z\_stream\_s, 3991
- avail\_out
  - z\_stream\_s, 3991
- available
  - decaf::internal::io::StandardInputStream, 3524
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2813
  - decaf::internal::net::ssl::openssl::OpenSSLSocketImpl, 2833
  - decaf::internal::net::tcp::TcpSocket, 3685
- decaf::internal::net::tcp::TcpSocketInputStream, 3692
- decaf::io::BlockingByteArrayInputStream, 802
- decaf::io::BufferedInputStream, 896
- decaf::io::ByteArrayInputStream, 988
- decaf::io::FilterInputStream, 1857
- decaf::io::InputStream, 2004
- decaf::io::PushbackInputStream, 3089
- decaf::net::SocketImpl, 3475
- decaf::util::zip::InflaterInputStream, 1998
- availablePermits
  - decaf::util::concurrent::Semaphore, 3285
- availableProcessors
  - decaf::lang::System, 3673
- await
  - decaf::util::concurrent::CountDownLatch, 1487–1489
  - decaf::util::concurrent::locks::Condition, 1222, 1223
  - decaf::util::concurrent::locks::Condition, 1223
- awaitTermination
  - decaf::util::concurrent::ExecutorService, 1834
- awaitUninterruptibly
  - decaf::util::concurrent::locks::Condition, 1225
- awaitUntil
  - decaf::util::concurrent::locks::Condition, 1226
- back
  - decaf::util::StlQueue, 3558, 3559
- inflate\_state, 1983
- BackupTransport
  - activemq::transport::failover::BackupTransport, 719
  - activemq::transport::failover::BackupTransportPool, 723
- BackupTransportPool
  - activemq::transport::failover::BackupTransportPool, 721
- BAD
  - zutil.h, 4425
- base\_dist
  - zutil.h, 4427
- base\_length
  - zutil.h, 4427
- trees.h, 4427



---

BaseCommand internal\_state, 2082  
   activemq::commands::BaseCommand, BIG\_STRING\_TYPE  
     724 activemq::util::PrimitiveValueNode, 2964  
 BaseCommandMarshaller BINARY\_MIME\_TYPE  
   activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, MQBlobMessage,  
     744 177  
   activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller,  
     765 decaf::internal::net::tcp::TcpSocket, 3685  
   activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 3296, 3297  
     731 decaf::net::Socket, 3451  
   activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 3475  
     738 BindException  
   activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 798, 799  
     751 bitCount  
   activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller, 3041  
     758 decaf::lang::Long, 2380  
 before bits  
   decaf::util::Date, 1635 code, 1154  
 beforeEnd inflate\_state, 1983  
   activemq::core::Synchronization, 3659 BL\_CODES  
 beforeMarshal deflate.h, 4419  
   activemq::commands::ActiveMQMapMessage, internal\_state, 2082  
     333  
   activemq::commands::ActiveMQTextMessage, internal\_state, 2082  
     632  
   activemq::commands::BaseDataStructure, internal\_state, 2082  
     794  
   activemq::commands::Message, 2480 block\_start  
   activemq::commands::WireFormatInfo, internal\_state, 2082  
     3915 BLOCKED  
   activemq::wireformat::MarshalAware, 2445 decaf::lang::Thread, 3710  
 beforeMessagesConsumed BlockingByteArrayInputStream  
   activemq::core::ActiveMQConsumer, 285 decaf::io::BlockingByteArrayInputStream,  
     801, 802  
 beforeUnmarshal Boolean  
   activemq::commands::BaseDataStructure, decaf::lang::Boolean, 812  
     794  
   activemq::wireformat::MarshalAware, 2445 BOOLEAN\_TYPE  
 BEGIN activemq::util::PrimitiveValueNode, 2963  
   activemq::wireformat::stomp::StompCommand, 3573  
     3573 activemq::commands::BooleanExpression, 816  
 begin BooleanStream  
   activemq::core::ActiveMQTransactionCoordinator, 689  
     689 activemq::wireformat::openwire::utils::BooleanStream,  
       819  
 BEST\_COMPRESSION booleanValue  
   decaf::util::zip::Deflater, 1681  
 BEST\_SPEED decaf::lang::Boolean, 812  
   decaf::util::zip::Deflater, 1681 boolValue  
 bi\_buf activemq::util::PrimitiveValueNode::PrimitiveValue,  
   internal\_state, 2082 2958  
 bi\_valid branchQualifier

---

activemq::commands::XATransactionId,	activemq::commands::DiscoveryEvent,
3964	1724
BrokenBarrierException	brokerOutTime
decaf::util::concurrent::BrokenBarrierException,	activemq::commands::Message, 2491
821, 822	brokerPath
BrokerError	activemq::commands::ConnectionInfo,
activemq::commands::BrokerError, 824	1330
BrokerException	activemq::commands::ConsumerInfo,
activemq::exceptions::BrokerException,	1433
828	activemq::commands::DestinationInfo,
BrokerId	1695
activemq::commands::BrokerId, 830	activemq::commands::Message, 2491
brokerId	activemq::commands::ProducerInfo, 3047
activemq::commands::BrokerInfo, 861	brokerSequenceId
BrokerIdMarshaller	activemq::commands::MessageId, 2627
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller,	brokerUploadUrl
841	activemq::commands::BrokerInfo, 862
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller,	brokerURL
853	activemq::commands::BrokerInfo, 862
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,	Browser
833	activemq::core::ActiveMQQueueBrowser,
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller,	460
837	browser
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,	activemq::commands::ConsumerInfo,
845	1433
activemq::wireformat::openwire::marshal::v6::BrokerIdMarshaller,	buf
849	decaf::util::zip::DeflaterOutputStream,
BrokerInfo	1686
activemq::commands::BrokerInfo, 858	buff
BrokerInfoMarshaller	decaf::util::zip::InflaterInputStream, 2001
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller,	Buffer
872	decaf::nio::Buffer, 889
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller,	buffer
884	decaf::io::DataOutputStream, 1550
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller,	BufferedInputStream
863	decaf::io::BufferedInputStream, 895, 896
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller,	BufferedOutputStream
868	decaf::io::BufferedOutputStream, 900
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller,	BufferOverflowException
876	decaf::nio::BufferOverflowException, 914,
activemq::wireformat::openwire::marshal::v6::BrokerInfoMarshaller,	915
880	BufferUnderflowException
brokerInTime	decaf::nio::BufferUnderflowException,
activemq::commands::Message, 2491	917, 918
brokerMasterConnector	buildIncomingCommands
activemq::commands::ConnectionInfo,	activemq::transport::mock::ResponseBuilder,
1330	3232
brokerName	activemq::wireformat::openwire::OpenWireResponseBuilder,
activemq::commands::BrokerInfo, 861	2855
	buildMessage

- decaf::lang::Exception, 1797
- buildResponse
  - activemq::transport::mock::ResponseBuilder, 1042
  - 3232
  - CachedProducer
  - activemq::wireformat::openwire::OpenWireResponseBuilder, 1045
  - 2855
  - call
- BUSY\_STATE
  - decaf::util::concurrent::Callable, 1052
- deflate.h, 4419
- cancel
  - decaf::util::concurrent::Future, 1930
  - decaf::util::Timer, 3733
  - decaf::util::TimerTask, 3744
- Byte
  - decaf::lang::Byte, 920
  - zconf.h, 4429
- BYTE\_ARRAY\_TYPE
  - CancellationException
  - activemq::util::PrimitiveValueNode, 2964
  - decaf::util::concurrent::CancellationException, 1053, 1054
- BYTE\_TYPE
  - activemq::util::PrimitiveValueNode, 2963
  - capacity
- ByteArrayAdapter
  - decaf::internal::util::ByteArrayAdapter, 932–935
  - cause
  - decaf::nio::Buffer, 889
- ByteArrayBuffer
  - decaf::internal::nio::ByteBuffer, 964, 965
  - ceil
  - decaf::lang::Math, 2459
- ByteArrayInputStream
  - decaf::io::ByteArrayInputStream, 987, 988
  - CertificateEncodingException
  - decaf::security::cert::CertificateEncodingException, 1060
- ByteArrayOutputStream
  - decaf::io::ByteArrayOutputStream, 993
  - CertificateException
  - decaf::security::cert::CertificateException, 1061, 1062
- byteArrayValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 2958
  - CertificateExpiredException
  - decaf::security::cert::CertificateExpiredException, 1063, 1064
- ByteBuffer
  - decaf::nio::ByteBuffer, 1000
  - CertificateNotYetValidException
  - decaf::security::cert::CertificateNotYetValidException, 1065, 1066
- Bytef
  - zconf.h, 4429
  - CertificateParsingException
  - decaf::security::cert::CertificateParsingException, 1067, 1068
- BYTES
  - activemq::wireformat::stomp::StompCommandConstants, 3573
  - CHAR\_CONST
  - activemq::util::PrimitiveValueNode, 2963
- bytesToInt
  - decaf::net::InetAddress, 1977
  - Character
  - decaf::lang::Character, 1071
- byteValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 2958
  - CharArrayBuffer
  - decaf::internal::nio::CharArrayBuffer, 1081, 1082
  - charAt
  - decaf::lang::CharSequence, 1108
  - decaf::lang::String, 3611
  - decaf::nio::CharBuffer, 1096
  - CharBuffer
  - decaf::nio::CharBuffer, 1092
  - charf
  - zconf.h, 4429
- decaf::lang::Byte, 920
- decaf::lang::Character, 1071
- decaf::lang::Double, 1753
- decaf::lang::Float, 1867
- decaf::lang::Integer, 2041
- decaf::lang::Long, 2380
- decaf::lang::Number, 2787
- decaf::lang::Short, 3382

- charValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue819
  - 2958
- CHECK
  - inflate.h, 4425
- check
  - inflate\_state, 1983
- checkClosed
  - decaf::io::InputStreamReader, 2014
  - decaf::io::OutputStreamWriter, 2865
  - decaf::net::ServerSocket, 3297
  - decaf::net::Socket, 3452
- checkConnectionFactory
  - activemq::cmsutil::CmsAccessor, 1124
- checkDestinationResolver
  - activemq::cmsutil::CmsDestinationAccessor, 1128
- CheckedInputStream
  - decaf::util::zip::CheckedInputStream, 1110
- CheckedOutputStream
  - decaf::util::zip::CheckedOutputStream, 1113
- checkMapsIsUnmarshalled
  - activemq::commands::ActiveMQMapMessage, 333
- checkResult
  - decaf::internal::net::tcp::TcpSocket, 3686
- checkShutdown
  - activemq::state::ConnectionState, 1359
  - activemq::state::SessionState, 3378
  - activemq::state::TransactionState, 3814
- checkValidity
  - decaf::security::cert::X509Certificate, 3959
- ClassCastException
  - decaf::lang::exceptions::ClassCastException, 1117, 1118
- ClassName
  - activemq::commands::BrokerError::StackTraceElement, 3521
- cleanup
  - decaf::internal::AprPool, 697
- clear
  - activemq::core::ActiveMQSessionExecutor, 504
  - activemq::core::MessageDispatchChannel, 2561
  - activemq::util::ActiveMQProperties, 450
  - activemq::util::PrimitiveValueNode, 2967
  - activemq::wireformat::openwire::utils::BooleanStream, 1136
  - cms::CMSProperties, 1136
  - decaf::internal::util::ByteArrayAdapter, 935
  - decaf::nio::Buffer, 890
  - decaf::util::AbstractCollection, 151
  - decaf::util::AbstractQueue, 166
  - decaf::util::Collection, 1158
  - decaf::util::concurrent::ConcurrentStlMap, 1207
  - decaf::util::concurrent::SynchronousQueue, 3662
  - decaf::util::Map, 2420
  - decaf::util::PriorityQueue, 2980
  - decaf::util::Properties, 3074
  - decaf::util::StlList, 3537
  - decaf::util::StlMap, 3547
  - decaf::util::StlQueue, 3559
  - decaf::util::StlSet, 3568
  - clearBody
    - activemq::commands::ActiveMQBytesMessage, 205
  - activemq::commands::ActiveMQMapMessage, 333
  - activemq::commands::ActiveMQMessageTemplate, 398
  - activemq::commands::ActiveMQStreamMessage, 509
  - activemq::commands::ActiveMQTextMessage, 632
  - cms::Message, 2497
  - clearMessagesInProgress
    - activemq::core::ActiveMQConsumer, 285
    - activemq::core::ActiveMQSession, 489
    - activemq::core::ActiveMQSessionExecutor, 504
  - clearProperties
    - activemq::commands::ActiveMQMessageTemplate, 3521
    - cms::Message, 2498
  - clearProperty
    - decaf::lang::System, 3674
  - CLIENT\_ACKNOWLEDGE
    - cms::Session, 3308
  - clientId
    - activemq::commands::ConnectionInfo, 1330
  - activemq::commands::JournalTopicAck, 2147

activemq::commands::RemoveSubscriptionInfo,	decaf::internal::net::ssl::openssl::OpenSSLParameters,
3169	2796
activemq::commands::SubscriptionInfo,	decaf::internal::net::ssl::openssl::OpenSSLSocketException,
3620	2824
clientMaster	decaf::io::EOFException, 1791
activemq::commands::ConnectionInfo,	decaf::io::InterruptedIOException, 2091
1330	decaf::io::IOException, 2105
clockSequence	decaf::io::UnsupportedEncodingException,
decaf::util::UUID, 3902	3849
clone	decaf::io::UTFDataFormatException, 3900
activemq::commands::ActiveMQBlobMessage,	decaf::lang::ArrayPointer, 700
174	decaf::lang::Exception, 1797
activemq::commands::ActiveMQBytesMessage,	decaf::lang::exceptions::ClassCastException,
205	1119
activemq::commands::ActiveMQMapMessage,	decaf::lang::exceptions::IllegalArgumentException,
333	1955
activemq::commands::ActiveMQMessage,	decaf::lang::exceptions::IllegalMonitorStateException,
369	1957
activemq::commands::ActiveMQObjectMessage,	decaf::lang::exceptions::IllegalStateException,
415	1961
activemq::commands::ActiveMQQueue,	decaf::lang::exceptions::IllegalThreadStateException,
454	1964
activemq::commands::ActiveMQStreamMessage,	decaf::lang::exceptions::IndexOutOfBoundsException,
509	1969
activemq::commands::ActiveMQTempQueue,	decaf::lang::exceptions::InterruptedException,
575	2088
activemq::commands::ActiveMQTempTopic,	decaf::lang::exceptions::InvalidStateException,
603	2102
activemq::commands::ActiveMQTextMessage,	decaf::lang::exceptions::NoSuchElementException,
632	2780
activemq::commands::ActiveMQTopic,	decaf::lang::exceptions::NullPointerException,
661	2786
activemq::core::policies::DefaultPrefetchPolicy,	decaf::lang::exceptions::NumberFormatException,
1641	2791
activemq::core::policies::DefaultRedeliveryPolicy,	decaf::lang::exceptions::RuntimeException,
1645	3269
activemq::core::PrefetchPolicy, 2926	decaf::lang::exceptions::UnsupportedOperationException,
activemq::core::RedeliveryPolicy, 3123	3851
activemq::exceptions::ActiveMQException,	decaf::lang::Throwable, 3725
329	decaf::net::BindException, 800
activemq::exceptions::BrokerException,	decaf::net::ConnectException, 1232
828	decaf::net::HttpRetryException, 1950
activemq::util::ActiveMQProperties, 450	decaf::net::MalformedURLException, 2418
activemq::wireformat::stomp::StompFrame,	decaf::net::NoRouteToHostException,
3578	2775
cms::BytesMessage, 1026	decaf::net::PortUnreachableException,
cms::CMSProperties, 1136	2924
cms::Destination, 1690	decaf::net::ProtocolException, 3085
cms::Message, 2498	decaf::net::SocketException, 3467
	decaf::net::SocketTimeoutException, 3489

decaf::net::UnknownHostException, 3844	activemq::commands::ActiveMQDestination, 296
decaf::net::UnknownServiceException, 3846	activemq::commands::ActiveMQMapMessage, 334
decaf::net::URISyntaxException, 3883	activemq::commands::ActiveMQMessage, 369
decaf::nio::BufferOverflowException, 916	activemq::commands::ActiveMQObjectMessage, 415
decaf::nio::BufferUnderflowException, 918	activemq::commands::ActiveMQQueue, 454
decaf::nio::InvalidMarkException, 2098	activemq::commands::ActiveMQStreamMessage, 509
decaf::nio::ReadOnlyBufferException, 3117	activemq::commands::ActiveMQTempDestination, 548
decaf::security::cert::CertificateEncodingException, 1060	activemq::commands::ActiveMQTempQueue, 575
decaf::security::cert::CertificateException, 1062	activemq::commands::ActiveMQTempTopic, 604
decaf::security::cert::CertificateExpiredException, 1064	activemq::commands::ActiveMQTextMessage, 633
decaf::security::cert::CertificateNotYetValidException, 1066	activemq::commands::ActiveMQTopic, 661
decaf::security::cert::CertificateParsingException, 1068	activemq::commands::BooleanExpression, 816
decaf::security::GeneralSecurityException, 1936	activemq::commands::BrokerError, 824
decaf::security::InvalidKeyException, 2096	activemq::commands::BrokerId, 830
decaf::security::KeyException, 2257	activemq::commands::BrokerInfo, 858
decaf::security::KeyManagementException, 2259	activemq::commands::ConnectionControl, 1238
decaf::security::NoSuchAlgorithmException, 2778	activemq::commands::ConnectionError, 1267
decaf::security::NoSuchProviderException, 2783	activemq::commands::ConnectionId, 1298
decaf::security::SignatureException, 3442	activemq::commands::ConnectionInfo, 1326
decaf::util::concurrent::BrokenBarrierException, 822	activemq::commands::ConsumerControl, 1370
decaf::util::concurrent::CancellationException, 1055	activemq::commands::ConsumerId, 1399
decaf::util::concurrent::ExecutionException, 1831	activemq::commands::ConsumerInfo, 1428
decaf::util::concurrent::RejectedExecutionException, 3136	activemq::commands::ControlCommand, 1460
decaf::util::concurrent::TimeoutException, 3730	activemq::commands::DataArrayResponse, 1494
decaf::util::Properties, 3074	activemq::commands::DataResponse, 1551
decaf::util::zip::DataFormatException, 1522	activemq::commands::DataStructure, 1628
decaf::util::zip::ZipException, 3993	activemq::commands::DestinationInfo, 174
cloneDataStructure	activemq::commands::DiscoveryEvent, 205
activemq::commands::ActiveMQBlobMessage, 174	
activemq::commands::ActiveMQBytesMessage, 1693	

- 1723
- activemq::commands::ExceptionResponse, 3962
- 1803 close
- activemq::commands::FlushCommand, 1042
- 1901
- activemq::commands::IntegerResponse, 1046
- 2055
- activemq::commands::JournalQueueAck, 2907
- 2117
- activemq::commands::JournalTopicAck, 549
- 2144
- activemq::commands::JournalTrace, 1241
- activemq::commands::JournalTransaction, 1373
- 2199
- activemq::commands::KeepAliveInfo, 250
- activemq::commands::LastPartialCommand, 286
- 2261
- activemq::commands::LocalTransactionId, 443
- 2308
- activemq::commands::Message, 458
- activemq::commands::MessageAck, 489
- activemq::commands::MessageDispatch, 504
- 2556
- activemq::commands::MessageDispatchNotification, 2561
- 2592
- activemq::commands::MessageId, 3234
- activemq::commands::MessagePull, 1838
- activemq::commands::NetworkBridgeFilter, 1965
- 2747
- activemq::commands::PartialCommand, 2107
- 2868
- activemq::commands::ProducerAck, 2726
- activemq::commands::ProducerId, 3697
- activemq::commands::ProducerInfo, 3829
- activemq::commands::RemoveInfo, 2852
- activemq::commands::RemoveSubscriptionInfo, 1120
- 3166
- activemq::commands::ReplayCommand, 1234
- 3195
- activemq::commands::Response, 3309
- activemq::commands::SessionId, 3523
- activemq::commands::SessionInfo, 3526
- activemq::commands::ShutdownInfo, 2813
- activemq::commands::SubscriptionInfo, 2834
- 3617
- activemq::commands::TransactionId, 3786
- activemq::commands::TransactionInfo, 3915
- activemq::commands::WireFormatInfo, 3915
- activemq::commands::XATransactionId, 3962
- activemq::cmsutil::CachedConsumer, 1042
- activemq::cmsutil::CachedProducer, 1046
- activemq::cmsutil::PooledSession, 2907
- activemq::commands::ActiveMQTempDestination, 549
- activemq::commands::ConnectionControl, 1241
- activemq::commands::ConsumerControl, 1373
- activemq::core::ActiveMQConnection, 250
- activemq::core::ActiveMQConsumer, 286
- activemq::core::ActiveMQProducer, 443
- activemq::core::ActiveMQQueueBrowser, 458
- activemq::core::ActiveMQSession, 489
- activemq::core::ActiveMQSessionExecutor, 504
- activemq::core::MessageDispatchChannel, 2561
- activemq::transport::correlator::ResponseCorrelator, 3234
- activemq::transport::failover::FailoverTransport, 1838
- activemq::transport::inactivity::InactivityMonitor, 1965
- activemq::transport::IOTransport, 2107
- activemq::transport::mock::MockTransport, 2726
- activemq::transport::tcp::TcpTransport, 3697
- activemq::transport::TransportFilter, 3829
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2852
- cms::Closeable, 1120
- cms::Connection, 1234
- cms::Session, 3309
- decaf::internal::io::StandardErrorOutputStream, 3523
- decaf::internal::io::StandardOutputStream, 3526
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2813
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2834

---

- decaf::internal::net::ssl::openssl::OpenSSL::SocketOutputStream, 1882
- 2836
- readShort, 1032
- decaf::internal::net::tcp::TcpSocket, 3686
- readString, 1033
- decaf::internal::net::tcp::TcpSocketInputStream, 3692
- readUnsignedShort, 1033
- readUTF, 1034
- decaf::internal::net::tcp::TcpSocketOutputStream, 3695
- write, 1034
- setBodyBytes, 1034
- decaf::BlockingByteArrayInputStream, 802
- writeBoolean, 1035
- writeByte, 1035
- decaf::io::BufferedInputStream, 897
- writeBytes, 1036
- decaf::io::Closeable, 1121
- writeChar, 1037
- decaf::io::FilterInputStream, 1857
- writeDouble, 1037
- decaf::io::FilterOutputStream, 1862
- writeFloat, 1037
- decaf::io::InputStream, 2004
- writeInt, 1038
- decaf::io::InputStreamReader, 2014
- writeLong, 1038
- decaf::io::OutputStream, 2858
- writeShort, 1039
- decaf::io::OutputStreamWriter, 2866
- writeString, 1039
- decaf::net::ServerSocket, 3297
- writeUnsignedShort, 1039
- decaf::net::Socket, 3452
- writeUTF, 1040
- decaf::net::SocketImpl, 3475
- cms::Closeable, 1119
- decaf::util::logging::ConsoleHandler, 1368
- ~Closeable, 1120
- decaf::util::logging::StreamHandler, 3593
- close, 1120
- decaf::util::zip::DeflaterOutputStream, 1685
- cms::CMSEException, 1130
- ~CMSEException, 1132
- decaf::util::zip::InflaterInputStream, 1998
- CMSEException, 1131, 1132
- CLOSE\_FAILURE
- getCause, 1132
- decaf::util::logging::ErrorManager, 1793
- getMessage, 1132
- closed
- getStackTrace, 1132
- decaf::io::FilterInputStream, 1860
- getStackTraceString, 1132
- decaf::io::FilterOutputStream, 1864
- printStackTrace, 1133
- CloseTransportsTask
- setMark, 1133
- activemq::transport::failover::CloseTransportsTask, 1122
- Task, 1133
- cms::CMSProperties, 1135
- ~CMSProperties, 1136
- cluster
- clear, 1136
- activemq::commands::Message, 2491
- clone, 1136
- cms, 122
- copy, 1136
- cms/Config.h
- CMS\_API, 4087
- getProperty, 1136, 1137
- cms::BytesMessage, 1023
- hasProperty, 1137
- ~BytesMessage, 1026
- isEmpty, 1137
- clone, 1026
- remove, 1137
- getBodyBytes, 1026
- setProperty, 1138
- getBodyLength, 1027
- toArray, 1138
- readBoolean, 1027
- toString, 1138
- readByte, 1028
- cms::CMSSecurityException, 1139
- ~CMSSecurityException, 1139
- readBytes, 1028, 1029
- CMSSecurityException, 1139
- readChar, 1030
- cms::Connection, 1232
- ~Connection, 1234
- readDouble, 1030
- close, 1234
- readFloat, 1031
- readInt, 1031

---



- createSession, 1234
- getClientID, 1235
- getExceptionListener, 1235
- getMetaData, 1235
- setClientID, 1236
- setExceptionListener, 1236
- cms::ConnectionFactory, 1294
  - ~ConnectionFactory, 1295
  - createCMSConnectionFactory, 1295
  - createConnection, 1295, 1296
- cms::ConnectionMetaData, 1355
  - ~ConnectionMetaData, 1356
  - getCMSMajorVersion, 1356
  - getCMSMinorVersion, 1356
  - getCMSProviderName, 1356
  - getCMSVersion, 1357
  - getCMSXPropertyNames, 1357
  - getProviderMajorVersion, 1357
  - getProviderMinorVersion, 1358
  - getProviderVersion, 1358
- cms::DeliveryMode, 1687
  - ~DeliveryMode, 1688
  - DELIVERY\_MODE, 1688
  - NON\_PERSISTENT, 1688
  - PERSISTENT, 1688
- cms::Destination, 1688
  - ~Destination, 1690
  - clone, 1690
  - copy, 1690
  - DestinationType, 1689
  - getCMSProperties, 1690
  - getDestinationType, 1690
  - QUEUE, 1689
  - TEMPORARY\_QUEUE, 1689
  - TEMPORARY\_TOPIC, 1689
  - TOPIC, 1689
- cms::ExceptionListener, 1801
  - ~ExceptionListener, 1801
  - onException, 1801
- cms::IllegalStateException, 1958
  - ~IllegalStateException, 1959
  - IllegalStateException, 1958, 1959
- cms::InvalidClientIDException, 2091
  - ~InvalidClientIDException, 2092
  - InvalidClientIDException, 2092
- cms::InvalidDestinationException, 2093
  - ~InvalidDestinationException, 2093
  - InvalidDestinationException, 2093
- cms::InvalidSelectorException, 2099
  - ~InvalidSelectorException, 2100
- InvalidSelectorException, 2100
- cms::MapMessage, 2431
  - ~MapMessage, 2434
  - getBoolean, 2434
  - getByte, 2434
  - getBytes, 2434
  - getChar, 2435
  - getDouble, 2435
  - getFloat, 2436
  - getInt, 2436
  - getLong, 2436
  - getMapNames, 2437
  - getShort, 2437
  - getString, 2438
  - itemExists, 2438
  - setBoolean, 2438
  - setByte, 2439
  - setBytes, 2439
  - setChar, 2440
  - setDouble, 2440
  - setFloat, 2441
  - setInt, 2441
  - setLong, 2441
  - setShort, 2442
  - setString, 2442
- cms::Message, 2493
  - ~Message, 2497
  - acknowledge, 2497
  - clearBody, 2497
  - clearProperties, 2498
  - clone, 2498
  - getBooleanProperty, 2498
  - getByteProperty, 2499
  - getCMSCorrelationID, 2500
  - getCMSDeliveryMode, 2500
  - getCMSDestination, 2501
  - getCMSExpiration, 2501
  - getCMSMessageID, 2502
  - getCMSPriority, 2503
  - getCMSRedelivered, 2503
  - getCMSReplyTo, 2504
  - getCMSTimestamp, 2504
  - getCMSType, 2505
  - getDoubleProperty, 2506
  - getFloatProperty, 2506
  - getIntProperty, 2507
  - getLongProperty, 2507
  - getPropertyNames, 2508
  - getShortProperty, 2508
  - getStringProperty, 2509

- propertyExists, 2510
- setBooleanProperty, 2510
- setByteProperty, 2511
- setCMSCorrelationID, 2511
- setCMSDeliveryMode, 2512
- setCMSDestination, 2513
- setCMSExpiration, 2513
- setCMSMessageID, 2514
- setCMSPriority, 2514
- setCMSRedelivered, 2515
- setCMSReplyTo, 2515
- setCMSTimestamp, 2516
- setCMSType, 2516
- setDoubleProperty, 2517
- setFloatProperty, 2518
- setIntProperty, 2518
- setLongProperty, 2519
- setShortProperty, 2519
- setStringProperty, 2520
- cms::MessageConsumer, 2550
  - ~MessageConsumer, 2551
  - getMessageListener, 2551
  - getMessageSelector, 2552
  - receive, 2552, 2553
  - receiveNoWait, 2553
  - setMessageListener, 2553
- cms::MessageEnumeration, 2620
  - ~MessageEnumeration, 2620
  - hasMoreMessages, 2620
  - nextMessage, 2621
- cms::MessageEOFException, 2621
  - ~MessageEOFException, 2622
  - MessageEOFException, 2622
- cms::MessageFormatException, 2622
  - ~MessageFormatException, 2623
  - MessageFormatException, 2623
- cms::MessageListener, 2652
  - ~MessageListener, 2652
  - onMessage, 2652
- cms::MessageNotReadableException, 2679
  - ~MessageNotReadableException, 2680
  - MessageNotReadableException, 2679, 2680
- cms::MessageNotWriteableException, 2680
  - ~MessageNotWriteableException, 2681
  - MessageNotWriteableException, 2681
- cms::MessageProducer, 2681
  - ~MessageProducer, 2683
  - getDeliveryMode, 2683
  - getDisableMessageID, 2683
  - getDisableMessageTimeStamp, 2683
  - getPriority, 2684
  - getTimeToLive, 2684
  - send, 2685–2687
  - setDeliveryMode, 2687
  - setDisableMessageID, 2688
  - setDisableMessageTimeStamp, 2688
  - setPriority, 2688
  - setTimeToLive, 2689
- cms::ObjectMessage, 2791
  - ~ObjectMessage, 2792
- cms::Queue, 3093
  - ~Queue, 3094
  - getQueueName, 3094
- cms::QueueBrowser, 3098
  - ~QueueBrowser, 3099
  - getEnumeration, 3099
  - getMessageSelector, 3099
  - getQueue, 3100
- cms::Session, 3305
  - ~Session, 3308
  - AcknowledgeMode, 3308
  - AUTO\_ACKNOWLEDGE, 3308
  - CLIENT\_ACKNOWLEDGE, 3308
  - close, 3309
  - commit, 3309
  - createBrowser, 3309, 3310
  - createBytesMessage, 3310
  - createConsumer, 3311, 3312
  - createDurableConsumer, 3313
  - createMapMessage, 3313
  - createMessage, 3314
  - createProducer, 3314
  - createQueue, 3314
  - createStreamMessage, 3315
  - createTemporaryQueue, 3315
  - createTemporaryTopic, 3316
  - createTextMessage, 3316
  - createTopic, 3316
  - DUPS\_OK\_ACKNOWLEDGE, 3308
  - getAcknowledgeMode, 3317
  - INDIVIDUAL\_ACKNOWLEDGE, 3308
  - isTransacted, 3317
  - recover, 3318
  - rollback, 3318
  - SESSION\_TRANSACTED, 3308
  - unsubscribe, 3318
- cms::Startable, 3527
  - ~Startable, 3527
  - start, 3527

- cms::Stoppable, 3590
  - ~Stoppable, 3591
  - stop, 3591
- cms::StreamMessage, 3595
  - ~StreamMessage, 3597
  - readBoolean, 3597
  - readByte, 3598
  - readBytes, 3598, 3599
  - readChar, 3600
  - readDouble, 3601
  - readFloat, 3601
  - readInt, 3602
  - readLong, 3602
  - readShort, 3603
  - readString, 3603
  - readUnsignedShort, 3604
  - writeBoolean, 3604
  - writeByte, 3605
  - writeBytes, 3605, 3606
  - writeChar, 3606
  - writeDouble, 3607
  - writeFloat, 3607
  - writeln, 3607
  - writeLong, 3608
  - writeShort, 3608
  - writeString, 3609
  - writeUnsignedShort, 3609
- cms::TemporaryQueue, 3701
  - ~TemporaryQueue, 3702
  - destroy, 3702
  - getQueueName, 3702
- cms::TemporaryTopic, 3703
  - ~TemporaryTopic, 3704
  - destroy, 3704
  - getTopicName, 3704
- cms::TextMessage, 3704
  - ~TextMessage, 3705
  - getText, 3705
  - setText, 3706
- cms::Topic, 3757
  - ~Topic, 3758
  - getTopicName, 3758
- cms::UnsupportedOperationException, 3852
  - ~UnsupportedOperationException, 3853
  - UnsupportedOperationException, 3853
- CMS\_API
  - cms/Config.h, 4087
- CmsAccessor
  - activemq::cmsutil::CmsAccessor, 1124
- CmsDestinationAccessor
  - activemq::cmsutil::CmsDestinationAccessor, 1128
- CMSException
  - cms::CMSException, 1131, 1132
- CMSExceptionSupport.h
  - AMQ\_CATCH\_ALL\_THROW\_CMSEXCEPTION, 4085
- CMSecurityException
  - cms::CMSecurityException, 1139
- CmsTemplate
  - activemq::cmsutil::CmsTemplate, 1143
- Code
  - deflate.h, 4419
- code, 1154
  - bits, 1154
  - ct\_data\_s, 1493
  - op, 1154
  - val, 1154
- CODELENS
  - inflate.h, 4424
- CODES
  - inftrees.h, 4426
- codes
  - inflate\_state, 1983
- codetype
  - inftrees.h, 4426
- comm\_max
  - gz\_header\_s, 1938
- command
  - activemq::commands::ControlCommand, 1462
- commandId
  - activemq::commands::PartialCommand, 2869
- COMMENT
  - inflate.h, 4424
- comment
  - gz\_header\_s, 1938
- COMMENT\_STATE
  - deflate.h, 4419
- COMMIT
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- commit
  - activemq::cmsutil::PooledSession, 2907
- activemq::core::ActiveMQConsumer, 286
- activemq::core::ActiveMQSession, 489
- activemq::core::ActiveMQTransactionContext, 689
- cms::Session, 3309

- compact
  - decaf::internal::nio::ByteBuffer, 970
  - decaf::internal::nio::CharArrayBuffer, 1084
  - decaf::internal::nio::DoubleArrayBuffer, 1769
  - decaf::internal::nio::FloatArrayBuffer, 1883
  - decaf::internal::nio::IntArrayBuffer, 2022
  - decaf::internal::nio::LongArrayBuffer, 2399
  - decaf::internal::nio::ShortArrayBuffer, 3397
  - decaf::nio::ByteBuffer, 1004
  - decaf::nio::CharBuffer, 1096
  - decaf::nio::DoubleBuffer, 1778
  - decaf::nio::FloatBuffer, 1891
  - decaf::nio::IntBuffer, 2030
  - decaf::nio::LongBuffer, 2407
  - decaf::nio::ShortBuffer, 3405
- COMPARATOR
  - activemq::commands::BrokerId, 830
  - activemq::commands::ConnectionId, 1298
  - activemq::commands::ConsumerId, 1399
  - activemq::commands::LocalTransactionId, 2308
  - activemq::commands::MessageId, 2625
  - activemq::commands::ProducerId, 3016
  - activemq::commands::SessionId, 3321
  - activemq::commands::TransactionId, 3760
  - activemq::commands::XATransactionId, 3961
- comparator
  - decaf::util::PriorityQueue, 2980
- compare
  - activemq::util::IdGenerator, 1951
  - decaf::lang::ArrayPointerComparator, 705
  - decaf::lang::Double, 1753
  - decaf::lang::Float, 1867
  - decaf::lang::PointerComparator, 2904
  - decaf::util::Comparator, 1190
  - decaf::util::comparators::Less, 2287
- compareAndSet
  - decaf::util::concurrent::atomic::AtomicBoolean, 706
  - decaf::util::concurrent::atomic::AtomicInteger, 710
  - decaf::util::concurrent::atomic::AtomicReference, 1207
- compareTo
  - activemq::commands::BrokerId, 830
  - activemq::commands::ConnectionId, 1298
- activemq::commands::ConsumerId, 1399
- activemq::commands::LocalTransactionId, 2308
- activemq::commands::MessageId, 2625
- activemq::commands::ProducerId, 3016
- activemq::commands::SessionId, 3322
- activemq::commands::TransactionId, 3761
- activemq::commands::XATransactionId, 3962
- decaf::lang::Boolean, 812
- decaf::lang::Byte, 921
- decaf::lang::Character, 1071
- decaf::lang::Comparable, 1187
- decaf::lang::Double, 1753, 1754
- decaf::lang::Float, 1868
- decaf::lang::Integer, 2042
- decaf::lang::Long, 2380, 2381
- decaf::lang::Short, 3383
- decaf::net::URI, 3857
- decaf::nio::ByteBuffer, 1005
- decaf::nio::CharBuffer, 1097
- decaf::nio::DoubleBuffer, 1778
- decaf::nio::FloatBuffer, 1892
- decaf::nio::IntBuffer, 2031
- decaf::nio::LongBuffer, 2408
- decaf::nio::ShortBuffer, 3405
- decaf::util::concurrent::TimeUnit, 3750
- decaf::util::Date, 1635
- decaf::util::logging::Level, 2292
- decaf::util::UUID, 3903
- COMPOSITE\_SEPARATOR
  - activemq::commands::ActiveMQDestination, 303
- CompositeData
  - activemq::util::CompositeData, 1192
- CompositeTaskRunner
  - activemq::threads::CompositeTaskRunner, 1195
- compressed
  - activemq::commands::Message, 2491
- Concurrent.h
  - asynchronized, 4511
  - WAIT\_INFINITE, 4511
- ConcurrentStlMap
  - decaf::util::concurrent::ConcurrentStlMap,
- condition
  - decaf::util::concurrent::ConditionHandle, 1227
- ConditionHandle
  - ConditionHandle

- decaf::util::concurrent::ConditionHandle, 1227
- CONFIG
  - decaf::util::logging::Level, 2294
- config
  - decaf::util::logging::Logger, 2349
- configure
  - activemq::core::PrefetchPolicy, 2926
  - activemq::core::RedeliveryPolicy, 3123
  - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 2450
  - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 2451
  - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 2448
  - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 2449
  - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 2449
  - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 2447
- configureSocket
  - activemq::transport::tcp::SslTransport, 3519
  - activemq::transport::tcp::TcpTransport, 3697
- CONNECT
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- connect
  - activemq::transport::tcp::TcpTransport, 3698
  - decaf::internal::net::ssl::openssl::OpenSslSocketException, 2813
  - decaf::internal::net::tcp::TcpSocket, 3686
  - decaf::net::Socket, 3452, 3453
  - decaf::net::SocketImpl, 3476
- CONNECTED
  - activemq::wireformat::stomp::StompCommandConstants, 3573
- connectedBrokers
  - activemq::commands::ConnectionControl, 1241
- ConnectException
  - decaf::net::ConnectException, 1230, 1231
- connection
  - activemq::commands::ActiveMQTempDestination, 550
  - activemq::commands::Message, 2491
- CONNECTION\_ADVISORY\_PREFIX
  - activemq::commands::ActiveMQDestination, 303
  - CONNECTION\_ALWAYS\_SYNC\_SEND
  - activemq::core::ActiveMQConstants, 281
  - CONNECTION\_CLOSE\_TIMEOUT
  - activemq::core::ActiveMQConstants, 281
  - CONNECTION\_DISPATCH\_ASYNC
  - activemq::core::ActiveMQConstants, 281
  - CONNECTION\_PRODUCER\_WINDOW\_SIZE
  - activemq::core::ActiveMQConstants, 281
  - CONNECTION\_SEND\_TIMEOUT
  - activemq::core::ActiveMQConstants, 281
  - CONNECTION\_USE\_ASYNC\_SEND
  - activemq::core::ActiveMQConstants, 281
  - CONNECTION\_USE\_COMPRESSION
  - activemq::core::ActiveMQConstants, 281
  - ConnectionControl
  - activemq::commands::ConnectionControl, 1238
  - ConnectionControlFactory
  - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1251
  - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1263
  - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1243
  - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1243
  - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1255
  - activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller, 1259
  - ConnectionError
  - activemq::commands::ConnectionError, 1267
  - ConnectionErrorMarshaller
  - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1283
  - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1271
  - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1275
  - activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1279
  - activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1287
  - activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller, 1291
  - ConnectionId
  - activemq::commands::ConnectionId, 1298

connectionId  
     activemq::commands::BrokerInfo, 862  
     activemq::commands::ConnectionError, 1269  
     activemq::commands::ConnectionInfo, 1330  
     activemq::commands::ConsumerId, 1400  
     activemq::commands::DestinationInfo, 1695  
     activemq::commands::LocalTransactionId, 2310  
     activemq::commands::ProducerId, 3018  
     activemq::commands::RemoveSubscriptionInfo, 3169  
     activemq::commands::SessionId, 3324  
     activemq::commands::TransactionInfo, 3789  
 ConnectionIdMarshaller  
     activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1314  
     activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1302  
     activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1306  
     activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1310  
     activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1318  
     activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller, 1322  
 ConnectionInfo  
     activemq::commands::ConnectionInfo, 1326  
 ConnectionInfoMarshaller  
     activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1344  
     activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1331  
     activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1336  
     activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1340  
     activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1348  
     activemq::wireformat::openwire::marshal::v6::ConnectionInfoMarshaller, 1352  
 connectionInterruptProcessingComplete  
     activemq::state::ConnectionStateTracker, 1363  
 ConnectionState  
     activemq::state::ConnectionState, 1359  
     ConnectionStateTracker  
     activemq::state::ConnectionStateTracker, 1362  
     ConsoleHandler  
     decaf::util::logging::ConsoleHandler, 1368  
     const  
     zconf.h, 4429  
     ConstReferenceType  
     decaf::lang::ArrayPointer, 699  
     CONSUMER\_ADVISORY\_PREFIX  
     activemq::commands::ActiveMQDestination, 303  
     CONSUMER\_DISPATCHASYNC  
     activemq::core::ActiveMQConstants, 280  
     CONSUMER\_EXCLUSIVE  
     activemq::core::ActiveMQConstants, 281  
     CONSUMER\_NOLOCAL  
     activemq::core::ActiveMQConstants, 280  
     CONSUMER\_PREFETCHSIZE  
     activemq::core::ActiveMQConstants, 280  
     CONSUMER\_PRIORITY  
     activemq::core::ActiveMQConstants, 281  
     CONSUMER\_RETROACTIVE  
     activemq::core::ActiveMQConstants, 280  
     CONSUMER\_SELECTOR  
     activemq::core::ActiveMQConstants, 281  
     ConsumerControl  
     activemq::core::ActiveMQConstants, 281  
     ConsumerControlMarshaller, 1370  
     ConsumerControlMarshaller  
     activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1377  
     activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1379  
     activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1379  
     activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1380  
     activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1380  
     activemq::wireformat::openwire::marshal::v6::ConsumerControlMarshaller, 1380  
     ConsumerId  
     activemq::wireformat::openwire::marshal::v6::ConsumerIdMarshaller, 1399  
     consumerId  
     activemq::commands::ConsumerControl, 1373  
     activemq::commands::ConsumerInfo, 1433

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

- activemq::wireformat::stomp::StompHelper, 3585
- COPY
  - gzguts.h, 4422
  - inflate.h, 4424
- copy
  - activemq::commands::ActiveMQQueue, 454
  - activemq::commands::ActiveMQTempQueue, 575
  - activemq::commands::ActiveMQTempTopic, 604
  - activemq::commands::ActiveMQTopic, 661
  - activemq::util::ActiveMQProperties, 450
  - activemq::wireformat::stomp::StompFrame, 3578
  - cms::CMSProperties, 1136
  - cms::Destination, 1690
  - decaf::util::AbstractCollection, 153
  - decaf::util::concurrent::ConcurrentStlMap, 1209
  - decaf::util::Map, 2423
  - decaf::util::Properties, 3075
  - decaf::util::StlList, 3538
  - decaf::util::StlMap, 3548, 3549
  - decaf::util::StlSet, 3569
- COPY\_
  - inflate.h, 4424
- copyDataStructure
  - activemq::commands::ActiveMQBlobMessage, 174
  - activemq::commands::ActiveMQBytesMessage, 205
  - activemq::commands::ActiveMQDestination, 296
  - activemq::commands::ActiveMQMapMessage, 334
  - activemq::commands::ActiveMQMessage, 370
  - activemq::commands::ActiveMQObjectMessage, 415
  - activemq::commands::ActiveMQQueue, 455
  - activemq::commands::ActiveMQStreamMessage, 510
  - activemq::commands::ActiveMQTempDestination, 549
  - activemq::commands::ActiveMQTempQueue, 576
  - activemq::commands::ActiveMQTempTopic, 604
  - activemq::commands::ActiveMQTextMessage, 633
  - activemq::commands::ActiveMQTopic, 661
  - activemq::commands::BaseCommand, 724
  - activemq::commands::BaseDataStructure, 795
  - activemq::commands::BooleanExpression, 817
  - activemq::commands::BrokerError, 824
  - activemq::commands::BrokerId, 830
  - activemq::commands::BrokerInfo, 858
  - activemq::commands::ConnectionControl, 1239
  - activemq::commands::ConnectionError, 1267
  - activemq::commands::ConnectionId, 1299
  - activemq::commands::ConnectionInfo, 1326
  - activemq::commands::ConsumerControl, 1370
  - activemq::commands::ConsumerId, 1399
  - activemq::commands::ConsumerInfo, 1428
  - activemq::commands::ControlCommand, 1460
  - activemq::commands::DataArrayResponse, 1494
  - activemq::commands::DataResponse, 1551
  - activemq::commands::DataStructure, 1629
  - activemq::commands::DestinationInfo, 1693
  - activemq::commands::DiscoveryEvent, 1723
  - activemq::commands::ExceptionResponse, 1803
  - activemq::commands::FlushCommand, 1901
  - activemq::commands::IntegerResponse, 2055
  - activemq::commands::JournalQueueAck, 2117
  - activemq::commands::JournalTopicAck, 2145
  - activemq::commands::JournalTrace, 2173



activemq::commands::JournalTransactionCounterType  
 2199  
 activemq::commands::KeepAliveInfo, 2226  
 activemq::commands::LastPartialCommandTokens  
 2261  
 activemq::commands::LocalTransactionCRC32  
 2308  
 activemq::commands::Message, 2481  
 activemq::commands::MessageAck, 2522  
 activemq::commands::MessageDispatch, 2556  
 activemq::commands::MessageDispatchNotification, 2592  
 activemq::commands::MessageId, 2626  
 activemq::commands::MessagePull, 2697  
 activemq::commands::NetworkBridgeFilter, 2747  
 activemq::commands::PartialCommand, 2868  
 activemq::commands::ProducerAck, 2985  
 activemq::commands::ProducerId, 3016  
 activemq::commands::ProducerInfo, 3044  
 activemq::commands::RemoveInfo, 3139  
 activemq::commands::RemoveSubscriptionInfo, 3167  
 activemq::commands::ReplayCommand, 3195  
 activemq::commands::Response, 3229  
 activemq::commands::SessionId, 3322  
 activemq::commands::SessionInfo, 3349  
 activemq::commands::ShutdownInfo, 3414  
 activemq::commands::SubscriptionInfo, 3617  
 activemq::commands::TransactionId, 3761  
 activemq::commands::TransactionInfo, 3786  
 activemq::commands::WireFormatInfo, 3915  
 activemq::commands::XATransactionId, 3962  
 correlationId  
 activemq::commands::Message, 2491  
 activemq::commands::MessagePull, 2699  
 activemq::commands::Response, 3231  
 countDown  
 decaf::util::concurrent::CountDownLatch, 1489  
 CountDownLatch  
 decaf::util::concurrent::CountDownLatch, 1487  
 decaf::lang::ArrayPointer, 699  
 decaf::lang::Pointer, 2898  
 decaf::util::StringTokenizer, 3614  
 decaf::util::zip::CRC32, 1491  
 crc32.h  
 crc\_table, 4417  
 crc\_table  
 crc32.h, 4417  
 activemq::transport::failover::FailoverTransportFactory, 1847  
 activemq::transport::mock::MockTransportFactory, 2734  
 activemq::transport::tcp::TcpTransportFactory, 3700  
 activemq::transport::TransportFactory, 3826  
 activemq::util::CMSExceptionSupport, 1134  
 decaf::internal::net::tcp::TcpSocket, 3686  
 decaf::internal::util::concurrent::ConditionImpl, 1228  
 decaf::internal::util::concurrent::MutexImpl, 2742  
 decaf::net::SocketImpl, 3476  
 decaf::net::URI, 3857  
 createBrowser  
 activemq::cmsutil::PooledSession, 2908  
 activemq::core::ActiveMQSession, 489, 490  
 cms::Session, 3309, 3310  
 createByteBuffer  
 decaf::internal::nio::BufferFactory, 903, 904  
 createBytesMessage  
 activemq::cmsutil::PooledSession, 2909  
 activemq::core::ActiveMQSession, 490, 491  
 cms::Session, 3310  
 createCachedConsumer  
 activemq::cmsutil::PooledSession, 2909  
 createCachedProducer  
 activemq::cmsutil::PooledSession, 2910  
 createCharBuffer  
 decaf::internal::nio::BufferFactory, 905, 906  
 createCMSConnectionFactory

cms::ConnectionFactory, 1295	activemq::util::CMSExceptionSupport, 1134
createComposite	activemq::util::CMSExceptionSupport, 1134
activemq::transport::failover::FailoverTransportFactory, 1848	activemq::wireformat::openwire::OpenWireFormat, 2840
activemq::transport::mock::MockTransportFactory, 2735	activemq::wireformat::stomp::StompWireFormat, 3587
activemq::transport::tcp::TcpTransportFactory, 3700	activemq::wireformat::WireFormat, 3908
activemq::transport::TransportFactory, 3826	createObject
createConnection	activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1578
activemq::cmsutil::CmsAccessor, 1125	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessage, 183
activemq::core::ActiveMQConnectionFactory, 267, 268	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessage, 225
cms::ConnectionFactory, 1295, 1296	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessage, 349
createConsumer	activemq::wireformat::openwire::marshal::v1::ActiveMQMessage, 376
activemq::cmsutil::PooledSession, 2910, 2911	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessage, 422
activemq::core::ActiveMQSession, 491, 492	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessage, 465
cms::Session, 3311, 3312	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessage, 528
createDestination	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 583
activemq::commands::ActiveMQDestination, 296	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 616
createDoubleBuffer	activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessage, 645
decaf::internal::nio::BufferFactory, 906, 907	activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMessage, 673
createDurableConsumer	activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 841
activemq::cmsutil::PooledSession, 2912	activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 872
activemq::core::ActiveMQSession, 492	activemq::wireformat::openwire::marshal::v1::ConnectionController, 1251
cms::Session, 3313	activemq::wireformat::openwire::marshal::v1::ConnectionError, 1283
createFloatBuffer	activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1314
decaf::internal::nio::BufferFactory, 908, 909	activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1344
createIntBuffer	activemq::wireformat::openwire::marshal::v1::ConsumerController, 1387
decaf::internal::nio::BufferFactory, 909, 910	
createLongBuffer	
decaf::internal::nio::BufferFactory, 911	
createMapMessage	
activemq::cmsutil::PooledSession, 2912	
activemq::core::ActiveMQSession, 493	
cms::Session, 3313	
createMessage	
activemq::cmsutil::MessageCreator, 2554	
activemq::cmsutil::PooledSession, 2913	
activemq::core::ActiveMQSession, 493	
cms::Session, 3314	
createMessageEOFException	

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

activemq::wireformat::openwire::marshal::v2::BrokenWireFormat 2600  
 884  
 activemq::wireformat::openwire::marshal::v2::ConversationControlMapBuilder 2629  
 1263  
 activemq::wireformat::openwire::marshal::v2::ConversationFormatMapBuilder 2701  
 1271  
 activemq::wireformat::openwire::marshal::v2::ConversationMarshaler 2750  
 1302  
 activemq::wireformat::openwire::marshal::v2::ConversationMarshaler 2875  
 1331  
 activemq::wireformat::openwire::marshal::v2::ConversationQueueIdMarshaller 2989  
 1375  
 activemq::wireformat::openwire::marshal::v2::ConversationWireFormatMapBuilder 3020  
 1403  
 activemq::wireformat::openwire::marshal::v2::ConversationWireFormatMapBuilder 3053  
 1435  
 activemq::wireformat::openwire::marshal::v2::ConversationWireFormatMapBuilder 3142  
 1464  
 activemq::wireformat::openwire::marshal::v2::DeleteArrayResponseMarshaller 3179  
 1497  
 activemq::wireformat::openwire::marshal::v2::DeleteResponseMarshaller 3206  
 1562  
 activemq::wireformat::openwire::marshal::v2::DeleteWireFormatMapBuilder 3242  
 1697  
 activemq::wireformat::openwire::marshal::v2::DiscoveryWireFormatMapBuilder 3325  
 1730  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller 3369  
 1810  
 activemq::wireformat::openwire::marshal::v2::FlushQueueWireFormatMapBuilder 3421  
 1908  
 activemq::wireformat::openwire::marshal::v2::GenerateResponseMarshaller 3641  
 2062  
 activemq::wireformat::openwire::marshal::v2::InactiveQueueAckMarshaller 3810  
 2124  
 activemq::wireformat::openwire::marshal::v2::InactiveTopicAckMarshaller 3932  
 2153  
 activemq::wireformat::openwire::marshal::v2::InactiveTraceMarshaller 3969  
 2175  
 activemq::wireformat::openwire::marshal::v2::InactiveTransactionalMarshaller 178  
 2206  
 activemq::wireformat::openwire::marshal::v2::KeepAliveWireFormatMapBuilder 221  
 2234  
 activemq::wireformat::openwire::marshal::v2::LastPartialFormatMarshaller 345  
 2272  
 activemq::wireformat::openwire::marshal::v2::LocalTransactionalMarshaller 372  
 2315  
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller 417  
 2531  
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller 461  
 2567

activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMessageMarshaller,	2210
524	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatQueueMarshaller,	2238
579	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2268
608	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2268
637	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2319
665	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2535
833	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2571
864	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2604
1243	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2641
1275	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2709
1306	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2762
1336	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2884
1379	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	2997
1407	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3028
1440	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3065
1468	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3150
1501	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3175
1566	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3210
1701	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3251
1734	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3341
1814	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3365
1912	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3433
2066	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3621
2132	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3798
2157	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3944
2179	
activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatTopicMarshaller;	3981

activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 187  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 1822  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 1916  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2070  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2136  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2165  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2187  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2218  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2242  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2280  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2327  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2539  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2579  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2608  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2633  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2713  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2766  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2888  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 2993  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 3024  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 3049  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 3162  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 3191  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 3198  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 3237  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageDispatcherMarshaller 3329

activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMarshaller, openwire::marshal::v5::ConsumerInfoMarshaller,  
 3373 1452  
 activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat, openwire::marshal::v5::ControlCommandMarshaller,  
 3437 1480  
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, openwire::marshal::v5::DataArrayResponseMarshaller,  
 3633 1513  
 activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMarshaller, openwire::marshal::v5::DataResponseMarshaller,  
 3806 1554  
 activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMarshaller, openwire::marshal::v5::DestinationInfoMarshaller,  
 3936 1717  
 activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMarshaller, openwire::marshal::v5::DiscoveryEventMarshaller,  
 3973 1746  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, openwire::marshal::v5::ExceptionResponseMarshaller,  
 195 1818  
 activemq::wireformat::openwire::marshal::v5::ActiveMQBinaryMessageMarshaller, openwire::marshal::v5::FlushCommandMarshaller,  
 233 1924  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, openwire::marshal::v5::IntegerResponseMarshaller,  
 357 2078  
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, openwire::marshal::v5::JournalQueueAckMarshaller,  
 384 2128  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v5::JournalTopicAckMarshaller,  
 430 2149  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v5::JournalTraceMarshaller,  
 473 2195  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v5::JournalTransactionMarshaller,  
 536 2214  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, openwire::marshal::v5::KeepAliveInfoMarshaller,  
 591 2246  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, openwire::marshal::v5::LastPartialCommandMarshaller,  
 620 2276  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, openwire::marshal::v5::LocalTransactionIdMarshaller,  
 649 2323  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, openwire::marshal::v5::MessageAckMarshaller,  
 677 2547  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat, openwire::marshal::v5::MessageDispatchMarshaller,  
 845 2575  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat, openwire::marshal::v5::MessageDispatchNotificationMarshaller,  
 876 2617  
 activemq::wireformat::openwire::marshal::v5::ActiveMQControlMarshaller, openwire::marshal::v5::MessageIdMarshaller,  
 1255 2637  
 activemq::wireformat::openwire::marshal::v5::ActiveMQControlMarshaller, openwire::marshal::v5::MessagePullMarshaller,  
 1287 2705  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller, openwire::marshal::v5::NetworkBridgeFilterMarshaller,  
 1318 2758  
 activemq::wireformat::openwire::marshal::v5::ActiveMQInfoMarshaller, openwire::marshal::v5::PartialCommandMarshaller,  
 1348 2880  
 activemq::wireformat::openwire::marshal::v5::ActiveMQControlMarshaller, openwire::marshal::v5::ProducerAckMarshaller,  
 1391 3001  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat, openwire::marshal::v5::ProducerIdMarshaller,  
 1419 3032

activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ConnectionControl  
 3061 1259  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ConnectionErrorM  
 3158 1291  
 activemq::wireformat::openwire::marshal::v5::ActiveMQSubscriptionInfoMarshaller,marshal::v6::ConnectionIdMar  
 3187 1322  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller,marshal::v6::ConnectionInfoM  
 3218 1352  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat,openwire::marshal::v6::ConsumerControl  
 3247 1395  
 activemq::wireformat::openwire::marshal::v5::ActiveMQIdMarshaller,openwire::marshal::v6::ConsumerIdMars  
 3337 1423  
 activemq::wireformat::openwire::marshal::v5::ActiveMQIdMarshaller,openwire::marshal::v6::ConsumerInfoMa  
 3357 1456  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ControlCommand  
 3429 1484  
 activemq::wireformat::openwire::marshal::v5::ActiveMQSubscriptionInfoMarshaller,marshal::v6::DataArrayRespon  
 3629 1517  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller,marshal::v6::DataResponseM  
 3790 1558  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::DestinationInfoM  
 3924 1713  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller,marshal::v6::DiscoveryEventM  
 3985 1726  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageMarshaller,marshal::v6::ExceptionRespon  
 199 1805  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageMarshaller,marshal::v6::FlushCommandM  
 237 1904  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller,marshal::v6::IntegerResponse  
 365 2058  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller,marshal::v6::JournalQueueAck  
 392 2120  
 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller,marshal::v6::JournalTopicAckl  
 438 2161  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller,marshal::v6::JournalTraceMar  
 481 2183  
 activemq::wireformat::openwire::marshal::v6::ActiveMQSerializedMessageMarshaller,marshal::v6::JournalTransactio  
 544 2203  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller,marshal::v6::KeepAliveInfoMar  
 599 2229  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller,marshal::v6::LastPartialComm  
 628 2263  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller,marshal::v6::LocalTransaction  
 653 2311  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller,marshal::v6::MessageAckMar  
 681 2527  
 activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat,openwire::marshal::v6::MessageDispatch  
 849 2587  
 activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat,openwire::marshal::v6::MessageDispatch  
 880 2596



activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, ConsumerInfo, 2645  
 1429  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, ProducerInfo, 3045  
 2721  
 activemq::wireformat::openwire::marshal::v6::MessagePushMarshaller, SessionInfo, 3350  
 2754  
 activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, createServerSocket, 1650–1651  
 2871  
 1650–1651  
 decaf::internal::net::DefaultServerSocketFactory, 1660–1661  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 3005  
 2805–2806  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 3036  
 decaf::net::ServerSocketFactory, 3302–3304  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, createSession, 3069  
 activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, activemq::cmsutil::CmsAccessor, 1125  
 3146  
 activemq::core::ActiveMQConnection, 250–251  
 activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, cms::Connection, 1234  
 3183  
 activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, createShortBuffer, 913  
 3214  
 decaf::internal::nio::BufferFactory, 912, 913  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, createSocket, 3261  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, activemq::transport::tcp::SslTransport, 3333  
 3519  
 activemq::transport::tcp::TcpTransport, 3353  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3698  
 decaf::internal::net::DefaultSocketFactory, 1654–1656  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, decaf::internal::net::ssl::DefaultSSLSocketFactory, 1666–1669  
 3637  
 decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2828–2831  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, decaf::net::SocketFactory, 3468–3470  
 3802  
 decaf::net::ssl::SSLSocketFactory, 3516  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, createSocketImpl, 3928  
 3965  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, decaf::net::SocketImplFactory, 3482  
 createProducer, createStreamMessage, 2914  
 activemq::cmsutil::PooledSession, 2913  
 activemq::core::ActiveMQSession, 493  
 cms::Session, 3315  
 createQueryString, createTemporaryName, 297  
 activemq::util::URISupport, 3878  
 activemq::commands::ActiveMQDestination, 297  
 createQueue, createTemporaryQueue, 2914  
 activemq::cmsutil::PooledSession, 2913  
 activemq::core::ActiveMQSession, 493  
 cms::Session, 3315  
 createRemoveCommand, createTemporaryTopic, 2914  
 activemq::commands::ConnectionInfo, 1327  
 activemq::core::ActiveMQSession, 494

cms::Session, 3316	activemq::commands::DataArrayResponse, 1496
createTextMessage	activemq::commands::DataResponse, 1552
activemq::cmsutil::PooledSession, 2915	activemq::commands::PartialCommand, 2869
activemq::core::ActiveMQSession, 495	
cms::Session, 3316	
createTopic	
activemq::cmsutil::PooledSession, 2915	data_type
activemq::core::ActiveMQSession, 495	z_stream_s, 3991
cms::Session, 3316	DataArrayResponse
createWireFormat	activemq::commands::DataArrayResponse, 1494
activemq::transport::AbstractTransportFactory, 171	DataArrayResponseMarshaller
activemq::wireformat::openwire::OpenWireFormatFactory, 2850	activemq::wireformat::openwire::marshal::v1::DataArrayResponse, 1509
activemq::wireformat::stomp::StompWireFormatFactory, 3590	activemq::wireformat::openwire::marshal::v2::DataArrayResponse, 1497
activemq::wireformat::WireFormatFactory, 3912	activemq::wireformat::openwire::marshal::v3::DataArrayResponse, 1501
criticalSection	activemq::wireformat::openwire::marshal::v4::DataArrayResponse, 1505
decaf::util::concurrent::ConditionHandle, 1227	activemq::wireformat::openwire::marshal::v5::DataArrayResponse, 1513
ct_data	activemq::wireformat::openwire::marshal::v6::DataArrayResponse, 1517
deflate.h, 4420	DataFormatException
ct_data_s, 1492	decaf::util::zip::DataFormatException, 1521, 1522
code, 1493	DataInputStream
dad, 1493	decaf::io::DataInputStream, 1534
dl, 1493	DataOutputStream
fc, 1493	decaf::io::DataOutputStream, 1548
freq, 1493	DataResponse
len, 1493	activemq::commands::DataResponse, 1551
CUNSUMER_MAXPENDINGMSGLIMIT	DataResponseMarshaller
activemq::core::ActiveMQConstants, 2860	activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1574
currentThread	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1562
decaf::lang::Thread, 3711	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1566
currentTimeMillis	activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1570
decaf::lang::System, 3674	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1554
	activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller, 1558
d_buf	
internal_state, 2082	
d_code	
deflate.h, 4419	
D_CODES	
deflate.h, 4420	
d_desc	
internal_state, 2082	
Dad	
deflate.h, 4420	
dad	
ct_data_s, 1493	dataStructure
data	activemq::commands::Message, 2491
	Date

- decaf::util::Date, 1634
- DAYS
- decaf::util::concurrent::TimeUnit, 3757
- DEBUG
- decaf::util::logging::Level, 2294
- Debug
- decaf::util::logging, 144
- debug
- decaf::util::logging::Logger, 2349
- decaf::util::logging::SimpleLogger, 3445
- decaf, 125
- decaf/lang/exceptions/ExceptionDefines.h
- DECAF\_CATCH\_EXCEPTION\_CONVERT, 4054
- DECAF\_CATCH\_NOTHROW, 4054
- DECAF\_CATCH\_RETHROW, 4054
- DECAF\_CATCHALL\_NOTHROW, 4055
- DECAF\_CATCHALL\_THROW, 4055
- decaf/util/Config.h
- DECAF\_API, 4087
- DECAF\_UNUSED, 4087
- HAVE\_PTHREAD\_H, 4087
- HAVE\_UUID\_T, 4087
- HAVE\_UUID\_UUID\_H, 4087
- decaf::internal, 125
- decaf::internal::AprPool, 696
  - ~AprPool, 696
  - AprPool, 696
  - cleanup, 697
  - getAprPool, 697
  - getGlobalPool, 697
- decaf::internal::DecafRuntime, 1637
  - ~DecafRuntime, 1638
  - DecafRuntime, 1638
  - getGlobalPool, 1638
- decaf::internal::io, 126
- decaf::internal::io::StandardErrorOutputStream, 3521
  - ~StandardErrorOutputStream, 3522
  - close, 3523
  - doWriteArrayBounded, 3523
  - doWriteByte, 3523
  - flush, 3523
  - StandardErrorOutputStream, 3522
- decaf::internal::io::StandardInputStream, 3524
  - ~StandardInputStream, 3524
  - available, 3524
  - doReadByte, 3525
  - StandardInputStream, 3524
- decaf::internal::io::StandardOutputStream, 3525
  - ~StandardOutputStream, 3526
  - close, 3526
  - doWriteArrayBounded, 3526
  - doWriteByte, 3526
  - flush, 3526
  - StandardOutputStream, 3526
- decaf::internal::net, 126
- decaf::internal::net::DefaultServerSocketFactory, 1648
  - ~DefaultServerSocketFactory, 1650
  - createServerSocket, 1650, 1651
  - DefaultServerSocketFactory, 1650
- decaf::internal::net::DefaultSocketFactory, 1652
  - ~DefaultSocketFactory, 1654
  - createSocket, 1654–1656
  - DefaultSocketFactory, 1654
- decaf::internal::net::Network, 2744
  - ~Network, 2745
  - addAsResource, 2745
  - addNetworkResource, 2745
  - getNetworkRuntime, 2745
  - getRuntimeLock, 2745
  - initializeNetworking, 2746
  - Network, 2745
  - shutdownNetworking, 2746
- decaf::internal::net::SocketFileDescriptor, 3471
  - ~SocketFileDescriptor, 3472
  - getValue, 3472
  - SocketFileDescriptor, 3472
- decaf::internal::net::ssl, 127
- decaf::internal::net::ssl::DefaultSSLContext, 1657
  - ~DefaultSSLContext, 1657
  - DefaultSSLContext, 1657
  - getContext, 1658
- decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1658
  - ~DefaultSSLServerSocketFactory, 1660
  - createServerSocket, 1660, 1661
  - DefaultSSLServerSocketFactory, 1660
  - getDefaultCipherSuites, 1662
  - getSupportedCipherSuites, 1662
- decaf::internal::net::ssl::DefaultSSLSocketFactory, 1663
  - ~DefaultSSLSocketFactory, 1666
  - createSocket, 1666–1669
  - DefaultSSLSocketFactory, 1666
  - getDefaultCipherSuites, 1669

- getSupportedCipherSuites, 1669
- decaf::internal::net::ssl::openssl, 127
- decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2792
  - ~OpenSSLContextSpi, 2793
  - OpenSSLContextSpi, 2793
  - OpenSSLSocket, 2795
  - OpenSSLSocketFactory, 2795
  - providerGetServerSocketFactory, 2793
  - providerGetSocketFactory, 2794
  - providerInit, 2794
- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2795
  - ~OpenSSLParameters, 2796
  - clone, 2796
  - getEnabledCipherSuites, 2796
  - getEnabledProtocols, 2796
  - getNeedClientAuth, 2796
  - getSupportedCipherSuites, 2796
  - getSupportedProtocols, 2796
  - getUseClientMode, 2796
  - getWantClientAuth, 2796
  - setEnabledCipherSuites, 2796
  - setEnabledProtocols, 2796
  - setNeedClientAuth, 2796
  - setUseClientMode, 2797
  - setWantClientAuth, 2797
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2797
  - ~OpenSSLServerSocket, 2799
  - accept, 2799
  - getEnabledCipherSuites, 2800
  - getEnabledProtocols, 2800
  - getNeedClientAuth, 2800
  - getSupportedCipherSuites, 2800
  - getSupportedProtocols, 2801
  - getWantClientAuth, 2801
  - OpenSSLServerSocket, 2799
  - setEnabledCipherSuites, 2801
  - setEnabledProtocols, 2802
  - setNeedClientAuth, 2802
  - setWantClientAuth, 2802
- decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2803
  - ~OpenSSLServerSocketFactory, 2805
  - createServerSocket, 2805, 2806
  - getDefaultCipherSuites, 2807
  - getSupportedCipherSuites, 2807
  - OpenSSLServerSocketFactory, 2805
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2808
  - OpenSSLSocket, 2813
  - available, 2813
  - close, 2813
  - connect, 2813
  - getEnabledCipherSuites, 2814
  - getEnabledProtocols, 2814
  - getInputStream, 2814
  - getNeedClientAuth, 2815
  - getOutputStream, 2815
  - getSupportedCipherSuites, 2816
  - getSupportedProtocols, 2816
  - getUseClientMode, 2816
  - getWantClientAuth, 2816
  - OpenSSLSocket, 2812, 2813
  - read, 2817
  - sendUrgentData, 2817
  - setEnabledCipherSuites, 2818
  - setEnabledProtocols, 2818
  - setNeedClientAuth, 2818
  - setOOBInline, 2819
  - setUseClientMode, 2819
  - setWantClientAuth, 2819
  - shutdownInput, 2820
  - shutdownOutput, 2820
  - startHandshake, 2820
  - write, 2821
- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2821
  - ~OpenSSLSocketException, 2824
  - clone, 2824
  - getErrorString, 2824
  - OpenSSLSocketException, 2822–2824
- decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2825
  - ~OpenSSLSocketFactory, 2828
  - createSocket, 2828–2831
  - getDefaultCipherSuites, 2831
  - getSupportedCipherSuites, 2831
  - OpenSSLSocketFactory, 2828
- decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2832
  - OpenSSLSocketInputStream, 2833
  - available, 2833
  - close, 2834
  - doReadArrayBounded, 2834
  - doReadByte, 2834
  - OpenSSLSocketInputStream, 2833
  - skip, 2834

- decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 3866
  - 2835
  - ~OpenSSLSocketOutputStream, 2836
  - close, 2836
  - doWriteArrayBounded, 2836
  - doWriteByte, 2836
  - OpenSSLSocketOutputStream, 2836
- decaf::internal::net::tcp, 128
- decaf::internal::net::tcp::TcpSocket, 3681
  - ~TcpSocket, 3685
  - accept, 3685
  - available, 3685
  - bind, 3685
  - checkResult, 3686
  - close, 3686
  - connect, 3686
  - create, 3686
  - getInputStream, 3687
  - getLocalAddress, 3687
  - getOption, 3687
  - getOutputStream, 3688
  - getSocketHandle, 3688
  - isClosed, 3688
  - isConnected, 3688
  - listen, 3688
  - read, 3689
  - setOption, 3689
  - shutdownInput, 3689
  - shutdownOutput, 3690
  - TcpSocket, 3685
  - write, 3690
- decaf::internal::net::tcp::TcpSocketInputStream, 3691
  - ~TcpSocketInputStream, 3692
  - available, 3692
  - close, 3692
  - doReadArrayBounded, 3693
  - doReadByte, 3693
  - skip, 3693
  - TcpSocketInputStream, 3692
- decaf::internal::net::tcp::TcpSocketOutputStream, 3694
  - ~TcpSocketOutputStream, 3695
  - close, 3695
  - doWriteArrayBounded, 3695
  - doWriteByte, 3695
  - TcpSocketOutputStream, 3695
- decaf::internal::net::URLEncoderDecoder, 3865
  - ~URLEncoderDecoder, 3866
  - decode, 3866
- decaf::internal::net::URIHelper, 3867
  - ~URIHelper, 3869
  - isValidDomainName, 3869
  - isValidHexChar, 3870
  - isValidHost, 3870
  - isValidIP4Word, 3870
  - isValidIP6Address, 3871
  - isValidIPv4Address, 3871
  - parseAuthority, 3871
  - parseURI, 3872
  - URIHelper, 3869
  - validateAuthority, 3872
  - validateFragment, 3872
  - validatePath, 3873
  - validateQuery, 3873
  - validateScheme, 3873
  - validateSsp, 3874
  - validateUserinfo, 3874
- decaf::internal::net::URIType, 3884
  - ~URIType, 3885
  - getAuthority, 3885
  - getFragment, 3886
  - getHost, 3886
  - getPath, 3886
  - getPort, 3886
  - getQuery, 3886
  - getScheme, 3886
  - getSchemeSpecificPart, 3887
  - getSource, 3887
  - getUserInfo, 3887
  - isAbsolute, 3887
  - isOpaque, 3887
  - isServerAuthority, 3888
  - isValid, 3888
  - setAbsolute, 3888
  - setAuthority, 3888
  - setFragment, 3888
  - setHost, 3889
  - setOpaque, 3889
  - setPath, 3889
  - setPort, 3889
  - setQuery, 3889
  - setScheme, 3890
  - setSchemeSpecificPart, 3890
  - setServerAuthority, 3890

- setSource, 3890
  - setUserInfo, 3891
  - setValid, 3891
  - URIType, 3885
- decaf::internal::nio, 128
- decaf::internal::nio::BufferFactory, 901
  - ~BufferFactory, 903
  - createByteBuffer, 903, 904
  - createCharBuffer, 905, 906
  - createDoubleBuffer, 906, 907
  - createFloatBuffer, 908, 909
  - createIntBuffer, 909, 910
  - createLongBuffer, 911
  - createShortBuffer, 912, 913
- decaf::internal::nio::ByteBuffer, 951
  - ~ByteBuffer, 966
  - array, 966
  - arrayOffset, 966
  - asCharBuffer, 967
  - asDoubleBuffer, 967
  - asFloatBuffer, 968
  - asIntBuffer, 968
  - asLongBuffer, 968
  - asReadOnlyBuffer, 969
  - asShortBuffer, 969
  - ByteBuffer, 964, 965
  - compact, 970
  - duplicate, 970
  - get, 970, 971
  - getChar, 971, 972
  - getDouble, 972
  - getFloat, 973
  - getInt, 973, 974
  - getLong, 974, 975
  - getShort, 975
  - hasArray, 976
  - isReadOnly, 976
  - put, 976, 977
  - putChar, 977, 978
  - putDouble, 978, 979
  - putFloat, 979, 980
  - putInt, 980, 981
  - putLong, 981, 982
  - putShort, 982, 983
  - setReadOnly, 983
  - slice, 984
- decaf::internal::nio::CharArrayBuffer, 1077
  - ~CharArrayBuffer, 1083
  - \_array, 1089
  - array, 1083
  - arrayOffset, 1083
  - asReadOnlyBuffer, 1084
  - CharArrayBuffer, 1081, 1082
  - compact, 1084
  - duplicate, 1085
  - get, 1085
  - hasArray, 1086
  - isReadOnly, 1086
  - length, 1089
  - offset, 1089
  - put, 1086, 1087
  - readOnly, 1089
  - setReadOnly, 1087
  - slice, 1088
  - subSequence, 1088
- decaf::internal::nio::DoubleArrayBuffer, 1762
  - ~DoubleArrayBuffer, 1768
  - array, 1768
  - arrayOffset, 1768
  - asReadOnlyBuffer, 1769
  - compact, 1769
  - DoubleArrayBuffer, 1766, 1767
  - duplicate, 1770
  - get, 1770
  - hasArray, 1771
  - isReadOnly, 1771
  - put, 1771, 1772
  - setReadOnly, 1772
  - slice, 1773
- decaf::internal::nio::FloatArrayBuffer, 1876
  - ~FloatArrayBuffer, 1882
  - array, 1882
  - arrayOffset, 1882
  - asReadOnlyBuffer, 1883
  - compact, 1883
  - duplicate, 1884
  - FloatArrayBuffer, 1880, 1881
  - get, 1884
  - hasArray, 1885
  - isReadOnly, 1885
  - put, 1885, 1886
  - setReadOnly, 1886
  - slice, 1887
- decaf::internal::nio::IntArrayBuffer, 2015
  - ~IntArrayBuffer, 2021
  - array, 2021
  - arrayOffset, 2021
  - asReadOnlyBuffer, 2022
  - compact, 2022
  - duplicate, 2023

- get, 2023
- hasArray, 2024
- IntArrayBuffer, 2019, 2020
- isReadOnly, 2024
- put, 2024, 2025
- setReadOnly, 2025
- slice, 2025
- decaf::internal::nio::LongArrayBuffer, 2392
  - ~LongArrayBuffer, 2397
  - array, 2398
  - arrayOffset, 2398
  - asReadOnlyBuffer, 2399
  - compact, 2399
  - duplicate, 2399
  - get, 2400
  - hasArray, 2401
  - isReadOnly, 2401
  - LongArrayBuffer, 2396, 2397
  - put, 2401, 2402
  - setReadOnly, 2402
  - slice, 2402
- decaf::internal::nio::ShortArrayBuffer, 3390
  - ~ShortArrayBuffer, 3395
  - array, 3395
  - arrayOffset, 3396
  - asReadOnlyBuffer, 3396
  - compact, 3397
  - duplicate, 3397
  - get, 3397, 3398
  - hasArray, 3398
  - isReadOnly, 3399
  - put, 3399
  - setReadOnly, 3400
  - ShortArrayBuffer, 3393–3395
  - slice, 3400
- decaf::internal::security, 129
- decaf::internal::security::SecureRandomImpl, 3275
  - ~SecureRandomImpl, 3276
  - providerGenerateSeed, 3276
  - providerNextBytes, 3276, 3277
  - providerSetSeed, 3277
  - SecureRandomImpl, 3276
- decaf::internal::util, 129
- decaf::internal::util::ByteArrayAdapter, 928
  - ~ByteArrayAdapter, 935
  - ByteArrayAdapter, 932–935
  - clear, 935
  - get, 935
  - getByteArray, 936
  - getCapacity, 936
  - getChar, 936
  - getCharArray, 937
  - getCharCapacity, 937
  - getDouble, 937
  - getDoubleArray, 937
  - getDoubleAt, 938
  - getDoubleCapacity, 938
  - getFloat, 938
  - getFloatArray, 939
  - getFloatAt, 939
  - getFloatCapacity, 939
  - getInt, 940
  - getIntArray, 940
  - getIntAt, 940
  - getIntCapacity, 941
  - getLong, 941
  - getLongArray, 941
  - getLongAt, 941
  - getLongCapacity, 942
  - getShort, 942
  - getShortArray, 943
  - getShortAt, 943
  - getShortCapacity, 943
  - operator[], 943, 944
  - put, 944
  - putChar, 944
  - putDouble, 945
  - putDoubleAt, 945
  - putFloat, 946
  - putFloatAt, 946
  - putInt, 946
  - putIntAt, 947
  - putLong, 947
  - putLongAt, 948
  - putShort, 948
  - putShortAt, 949
  - read, 949
  - resize, 950
  - write, 950
- decaf::internal::util::concurrent, 129
- decaf::internal::util::concurrent::ConditionImpl, 1228
  - create, 1228
  - destroy, 1228
  - notify, 1229
  - notifyAll, 1229
  - wait, 1229
- decaf::internal::util::concurrent::MutexImpl, 2742

- create, 2742
- destroy, 2742
- lock, 2743
- trylock, 2743
- unlock, 2743
- decaf::internal::util::concurrent::SynchronizableImpl, 3655
  - ~SynchronizableImpl, 3656
  - lock, 3656
  - notify, 3656
  - notifyAll, 3656
  - SynchronizableImpl, 3656
  - tryLock, 3657
  - unlock, 3657
  - wait, 3657, 3658
- decaf::internal::util::concurrent::Transferer, 3815
- decaf::internal::util::concurrent::TransferQueue, 3815
  - ~TransferQueue, 3816
  - transfer, 3816, 3817
  - TransferQueue, 3816
- decaf::internal::util::concurrent::TransferStack, 3817
  - ~TransferStack, 3818
  - transfer, 3818
  - TransferStack, 3818
- decaf::internal::util::GenericResource, 1937
  - ~GenericResource, 1938
  - GenericResource, 1938
  - getManaged, 1938
  - setManaged, 1938
- decaf::internal::util::HexStringParser, 1945
  - ~HexStringParser, 1946
  - HexStringParser, 1946
  - parse, 1946
  - parseDouble, 1946
  - parseFloat, 1946
- decaf::internal::util::Resource, 3223
  - ~Resource, 3223
- decaf::internal::util::ResourceLifecycleManager, 3224
  - ~ResourceLifecycleManager, 3224
  - addResource, 3224
  - destroyResources, 3224
  - ResourceLifecycleManager, 3224
- decaf::internal::util::TimerTaskHeap, 3745
  - ~TimerTaskHeap, 3746
  - adjustMinimum, 3746
- decaf::util::TimerTask, 3745
- deleteIfCancelled, 3746
- find, 3746
- insert, 3746
- isEmpty, 3747
- peek, 3747
- remove, 3747
- reset, 3747
- size, 3747
- TimerTaskHeap, 3746
- decaf::io, 130
- decaf::io::BlockingByteArrayInputStream, 800
  - ~BlockingByteArrayInputStream, 802
  - available, 802
  - BlockingByteArrayInputStream, 801, 802
  - close, 802
  - doReadArrayBounded, 802
  - doReadByte, 803
  - setByteArray, 803
  - skip, 803
- decaf::io::BufferedInputStream, 893
  - ~BufferedInputStream, 896
  - available, 896
  - BufferedInputStream, 895, 896
  - close, 897
  - doReadArrayBounded, 897
  - doReadByte, 897
  - mark, 897
  - markSupported, 897
  - reset, 898
  - skip, 898
- decaf::io::BufferedOutputStream, 899
  - ~BufferedOutputStream, 900
  - BufferedOutputStream, 900
  - doWriteArray, 900
  - doWriteArrayBounded, 901
  - doWriteByte, 901
  - flush, 901
- decaf::io::ByteArrayInputStream, 984
  - ~ByteArrayInputStream, 988
  - available, 988
  - ByteArrayInputStream, 987, 988
  - doReadArrayBounded, 989
  - doReadByte, 989
  - mark, 989
  - markSupported, 989
  - reset, 990
  - setByteArray, 990, 991
  - skip, 991
- decaf::io::ByteArrayOutputStream, 992
  - ~ByteArrayOutputStream, 993



- ByteArrayOutputStream, 993
- doWriteArrayBounded, 993
- doWriteByte, 993
- reset, 994
- size, 994
- toArray, 994
- toString, 994
- writeTo, 994
- decaf::io::Closeable, 1120
  - ~Closeable, 1121
  - close, 1121
- decaf::io::DataInput, 1523
  - ~DataInput, 1524
  - readBoolean, 1524
  - readByte, 1525
  - readChar, 1525
  - readDouble, 1525
  - readFloat, 1526
  - readFully, 1526, 1527
  - readInt, 1528
  - readLine, 1528
  - readLong, 1529
  - readShort, 1529
  - readString, 1529
  - readUnsignedByte, 1530
  - readUnsignedShort, 1530
  - readUTF, 1531
  - skipBytes, 1531
- decaf::io::DataInputStream, 1532
  - ~DataInputStream, 1534
  - DataInputStream, 1534
  - readBoolean, 1534
  - readByte, 1534
  - readChar, 1535
  - readDouble, 1535
  - readFloat, 1535
  - readFully, 1536
  - readInt, 1537
  - readLine, 1537
  - readLong, 1538
  - readShort, 1538
  - readString, 1539
  - readUnsignedByte, 1539
  - readUnsignedShort, 1539
  - readUTF, 1540
  - skipBytes, 1540
- decaf::io::DataOutput, 1541
  - ~DataOutput, 1542
  - writeBoolean, 1542
  - writeByte, 1543
  - writeBytes, 1543
  - writeChar, 1543
  - writeChars, 1544
  - writeDouble, 1544
  - writeFloat, 1544
  - writeInt, 1545
  - writeLong, 1545
  - writeShort, 1545
  - writeUnsignedShort, 1546
  - writeUTF, 1546
- decaf::io::DataOutputStream, 1546
  - ~DataOutputStream, 1548
  - buffer, 1550
  - DataOutputStream, 1548
  - doWriteArrayBounded, 1548
  - doWriteByte, 1548
  - size, 1549
  - writeBoolean, 1549
  - writeByte, 1549
  - writeBytes, 1549
  - writeChar, 1549
  - writeChars, 1549
  - writeDouble, 1549
  - writeFloat, 1549
  - writeInt, 1549
  - writeLong, 1549
  - writeShort, 1549
  - writeUnsignedShort, 1549
  - writeUTF, 1550
  - written, 1550
- decaf::io::EOFException, 1789
  - ~EOFException, 1791
  - clone, 1791
  - EOFException, 1790, 1791
- decaf::io::FileDescriptor, 1850
  - ~FileDescriptor, 1852
  - descriptor, 1852
  - err, 1852
  - FileDescriptor, 1852
  - in, 1852
  - out, 1852
  - readonly, 1853
  - sync, 1852
  - valid, 1852
- decaf::io::FilterInputStream, 1854
  - ~FilterInputStream, 1856
  - available, 1857
  - close, 1857
  - closed, 1860
  - doReadArray, 1857

- doReadArrayBounded, 1857
- doReadByte, 1858
- FilterInputStream, 1856
- inputStream, 1860
- isClosed, 1858
- mark, 1858
- markSupported, 1858
- own, 1860
- reset, 1859
- skip, 1859
- decaf::io::FilterOutputStream, 1861
  - ~FilterOutputStream, 1862
  - close, 1862
  - closed, 1864
  - doWriteArray, 1863
  - doWriteArrayBounded, 1863
  - doWriteByte, 1863
  - FilterOutputStream, 1862
  - flush, 1863
  - isClosed, 1864
  - outputStream, 1864
  - own, 1864
  - toString, 1864
- decaf::io::Flushable, 1899
  - ~Flushable, 1900
  - flush, 1900
- decaf::io::InputStream, 2002
  - ~InputStream, 2004
  - available, 2004
  - close, 2004
  - doReadArray, 2005
  - doReadArrayBounded, 2005
  - doReadByte, 2005
  - InputStream, 2004
  - lock, 2005
  - mark, 2006
  - markSupported, 2006
  - notify, 2006
  - notifyAll, 2007
  - read, 2007, 2008
  - reset, 2009
  - skip, 2010
  - toString, 2010
  - tryLock, 2011
  - unlock, 2011
  - wait, 2011, 2012
- decaf::io::InputStreamReader, 2013
  - ~InputStreamReader, 2014
  - checkClosed, 2014
  - close, 2014
  - doReadArrayBounded, 2015
  - InputStreamReader, 2014
  - ready, 2015
- decaf::io::InterruptedException, 2089
  - ~InterruptedException, 2091
  - clone, 2091
  - InterruptedException, 2090, 2091
- decaf::io::IOException, 2103
  - ~IOException, 2104
  - clone, 2105
  - IOException, 2103, 2104
- decaf::io::OutputStream, 2856
  - ~OutputStream, 2858
  - close, 2858
  - doWriteArray, 2858
  - doWriteArrayBounded, 2858
  - doWriteByte, 2859
  - flush, 2859
  - lock, 2859
  - notify, 2860
  - notifyAll, 2860
  - OutputStream, 2858
  - toString, 2860
  - tryLock, 2860
  - unlock, 2861
  - wait, 2861, 2862
  - write, 2863
- decaf::io::OutputStreamWriter, 2864
  - ~OutputStreamWriter, 2865
  - checkClosed, 2865
  - close, 2866
  - doWriteArrayBounded, 2866
  - flush, 2866
  - OutputStreamWriter, 2865
- decaf::io::PushbackInputStream, 3086
  - ~PushbackInputStream, 3089
  - available, 3089
  - doReadArrayBounded, 3090
  - doReadByte, 3090
  - mark, 3090
  - markSupported, 3090
  - PushbackInputStream, 3088, 3089
  - reset, 3091
  - skip, 3091
  - unread, 3092, 3093
- decaf::io::Reader, 3108
  - ~Reader, 3110
  - doReadArray, 3110
  - doReadArrayBounded, 3110
  - doReadChar, 3110

- doReadCharBuffer, 3110
- doReadVector, 3110
- mark, 3110
- markSupported, 3111
- read, 3111–3113
- Reader, 3110
- ready, 3113
- reset, 3114
- skip, 3114
- decaf::io::UnsupportedEncodingException, 3847
  - ~UnsupportedEncodingException, 3849
  - clone, 3849
  - UnsupportedEncodingException, 3847, 3848
- decaf::io::UTFDataFormatException, 3897
  - ~UTFDataFormatException, 3899
  - clone, 3900
  - UTFDataFormatException, 3898, 3899
- decaf::io::Writer, 3951
  - ~Writer, 3952
  - append, 3952, 3953
  - doAppendChar, 3954
  - doAppendCharSequence, 3954
  - doAppendCharSequenceStartEnd, 3954
  - doWriteArray, 3954
  - doWriteArrayBounded, 3954
  - doWriteChar, 3954
  - doWriteString, 3955
  - doWriteStringBounded, 3955
  - doWriteVector, 3955
  - write, 3955, 3956
  - Writer, 3952
- decaf::lang, 131
  - operator==, 133, 134
- decaf::lang::Appendable, 693
  - ~Appendable, 694
  - append, 694, 695
- decaf::lang::ArrayPointer, 697
  - ~ArrayPointer, 700
  - ArrayPointer, 699, 700
  - clone, 700
  - ConstReferenceType, 699
  - CounterType, 699
  - get, 701
  - length, 701
  - operator=, 702
  - operator==, 702, 704
  - operator[], 702
  - PointerType, 699
- ReferenceType, 699
- release, 702
- reset, 703
- swap, 703
- decaf::lang::ArrayPointerComparator, 704
  - compare, 705
  - operator(), 705
- decaf::lang::Boolean, 810
  - ~Boolean, 812
  - \_FALSE, 815
  - \_TRUE, 815
  - Boolean, 812
  - booleanValue, 812
  - compareTo, 812
  - equals, 813
  - operator<, 813
  - operator==, 814
  - parseBoolean, 814
  - toString, 815
  - valueOf, 815
- decaf::lang::Byte, 918
  - ~Byte, 920
  - Byte, 920
  - byteValue, 920
  - compareTo, 921
  - decode, 921
  - doubleValue, 922
  - equals, 922
  - floatValue, 922
  - intValue, 923
  - longValue, 923
  - MAX\_VALUE, 927
  - MIN\_VALUE, 927
  - operator<, 923
  - operator==, 924
  - parseByte, 924, 925
  - shortValue, 926
  - SIZE, 928
  - toString, 926
  - valueOf, 926, 927
- decaf::lang::Character, 1069
  - byteValue, 1071
  - Character, 1071
  - compareTo, 1071
  - digit, 1072
  - doubleValue, 1072
  - equals, 1072
  - floatValue, 1073
  - intValue, 1073
  - isDigit, 1073

- isISOControl, 1073
- isLetter, 1073
- isLetterOrDigit, 1074
- isLowerCase, 1074
- isUpperCase, 1074
- isWhitespace, 1074
- longValue, 1074
- MAX\_RADIX, 1076
- MAX\_VALUE, 1076
- MIN\_RADIX, 1076
- MIN\_VALUE, 1076
- operator<, 1074, 1075
- operator==, 1075
- shortValue, 1076
- SIZE, 1077
- toString, 1076
- valueOf, 1076
- decaf::lang::CharSequence, 1107
  - ~CharSequence, 1108
  - charAt, 1108
  - length, 1108
  - subSequence, 1108
  - toString, 1109
- decaf::lang::Comparable, 1186
  - ~Comparable, 1187
  - compareTo, 1187
  - equals, 1188
  - operator<, 1188
  - operator==, 1188
- decaf::lang::Double, 1751
  - ~Double, 1753
  - byteValue, 1753
  - compare, 1753
  - compareTo, 1753, 1754
  - Double, 1753
  - doubleToLongBits, 1754
  - doubleToRawLongBits, 1755
  - doubleValue, 1755
  - equals, 1755, 1756
  - floatValue, 1756
  - intValue, 1756
  - isInfinite, 1756
  - isNaN, 1757
  - longBitsToDouble, 1757
  - longValue, 1757
  - MAX\_VALUE, 1762
  - MIN\_VALUE, 1762
  - NaN, 1762
  - NEGATIVE\_INFINITY, 1762
  - operator<, 1758
  - operator==, 1758, 1759
  - parseDouble, 1759
  - POSITIVE\_INFINITY, 1762
  - shortValue, 1759
  - SIZE, 1762
  - toHexString, 1760
  - toString, 1760
  - valueOf, 1761
- decaf::lang::DYNAMIC\_CAST\_TOKEN, 1786
- decaf::lang::Exception, 1794
  - ~Exception, 1796
  - buildMessage, 1797
  - cause, 1800
  - clone, 1797
  - Exception, 1795, 1796
  - getCause, 1797
  - getMessage, 1798
  - getStackTrace, 1798
  - getStackTraceString, 1798
  - initCause, 1799
  - message, 1800
  - operator=, 1799
  - printStackTrace, 1799
  - setMark, 1799
  - setMessage, 1800
  - setStackTrace, 1800
  - stackTrace, 1800
  - what, 1800
- decaf::lang::exceptions, 134
- decaf::lang::exceptions::ClassCastException, 1117
  - ~ClassCastException, 1119
  - ClassCastException, 1117, 1118
  - clone, 1119
- decaf::lang::exceptions::IllegalArgumentException, 1953
  - ~IllegalArgumentException, 1955
  - clone, 1955
  - IllegalArgumentException, 1953, 1954
- decaf::lang::exceptions::IllegalMonitorStateException, 1955
  - ~IllegalMonitorStateException, 1957
  - clone, 1957
  - IllegalMonitorStateException, 1956, 1957
- decaf::lang::exceptions::IllegalStateException, 1959
  - ~IllegalStateException, 1961
  - clone, 1961
  - IllegalStateException, 1960, 1961

decaf::lang::exceptions::IllegalThreadStateException, 1962  
     ~IllegalThreadStateException, 1964  
     clone, 1964  
     IllegalThreadStateException, 1962, 1963  
 decaf::lang::exceptions::IndexOutOfBoundsException, 1967  
     ~IndexOutOfBoundsException, 1969  
     clone, 1969  
     IndexOutOfBoundsException, 1968, 1969  
 decaf::lang::exceptions::InterruptedException, 2086  
     ~InterruptedException, 2088  
     clone, 2088  
     InterruptedException, 2087, 2088  
 decaf::lang::exceptions::InvalidStateException, 2100  
     ~InvalidStateException, 2102  
     clone, 2102  
     InvalidStateException, 2101, 2102  
 decaf::lang::exceptions::NoSuchElementException, 2778  
     ~NoSuchElementException, 2780  
     clone, 2780  
     NoSuchElementException, 2779, 2780  
 decaf::lang::exceptions::NullPointerException, 2783  
     ~NullPointerException, 2785  
     clone, 2786  
     NullPointerException, 2784, 2785  
 decaf::lang::exceptions::NumberFormatException, 2789  
     ~NumberFormatException, 2791  
     clone, 2791  
     NumberFormatException, 2789, 2790  
 decaf::lang::exceptions::RuntimeException, 3267  
     ~RuntimeException, 3269  
     clone, 3269  
     RuntimeException, 3268, 3269  
 decaf::lang::exceptions::UnsupportedOperationException, 3849  
     ~UnsupportedOperationException, 3851  
     clone, 3851  
     UnsupportedOperationException, 3850, 3851  
 decaf::lang::Float, 1865  
     ~Float, 1867  
     byteValue, 1867  
     compare, 1867  
     compareTo, 1868  
     doubleValue, 1868  
     equals, 1869  
     Float, 1867  
     floatToIntBits, 1869  
     floatToRawIntBits, 1869  
     floatValue, 1870  
     intBitsToFloat, 1870  
     intValue, 1871  
     isInfinite, 1871  
     isNaN, 1871  
     longValue, 1872  
     MAX\_VALUE, 1876  
     MIN\_VALUE, 1876  
     NaN, 1876  
     NEGATIVE\_INFINITY, 1876  
     operator<, 1872  
     operator==, 1872, 1873  
     parseFloat, 1873  
     POSITIVE\_INFINITY, 1876  
     shortValue, 1873  
     SIZE, 1876  
     toHexString, 1874  
     toString, 1874  
     valueOf, 1875  
 decaf::lang::Integer, 2038  
     ~Integer, 2041  
     bitCount, 2041  
     byteValue, 2041  
     compareTo, 2042  
     decode, 2042  
     doubleValue, 2043  
     equals, 2043  
     floatValue, 2044  
     highestOneBit, 2044  
     Integer, 2041  
     intValue, 2044  
     longValue, 2044  
     lowestOneBit, 2045  
     MAX\_VALUE, 2054  
     MIN\_VALUE, 2054  
     numberOfLeadingZeros, 2045  
     numberOfTrailingZeros, 2045  
     operator<, 2046  
     operator==, 2046, 2047  
     parseInt, 2047, 2048  
     reverse, 2048  
     reverseBytes, 2048  
     rotateLeft, 2049  
     rotateRight, 2049

- shortValue, 2049
- signum, 2050
- SIZE, 2054
- toBinaryString, 2050
- toHexString, 2050
- toOctalString, 2051
- toString, 2051, 2052
- valueOf, 2052, 2053
- decaf::lang::Iterable, 2112
  - ~Iterable, 2113
  - iterator, 2113, 2114
- decaf::lang::Long, 2377
  - ~Long, 2380
  - bitCount, 2380
  - byteValue, 2380
  - compareTo, 2380, 2381
  - decode, 2381
  - doubleValue, 2382
  - equals, 2382
  - floatValue, 2382
  - highestOneBit, 2383
  - intValue, 2383
  - Long, 2379
  - longValue, 2383
  - lowestOneBit, 2383
  - MAX\_VALUE, 2392
  - MIN\_VALUE, 2392
  - numberOfLeadingZeros, 2384
  - numberOfTrailingZeros, 2384
  - operator<, 2384, 2385
  - operator==, 2385
  - parseLong, 2386
  - reverse, 2387
  - reverseBytes, 2387
  - rotateLeft, 2387
  - rotateRight, 2388
  - shortValue, 2388
  - signum, 2388
  - SIZE, 2392
  - toBinaryString, 2389
  - toHexString, 2389
  - toOctalString, 2390
  - toString, 2390
  - valueOf, 2390, 2391
- decaf::lang::Math, 2455
  - ~Math, 2457
  - abs, 2457, 2458
  - ceil, 2459
  - E, 2472
  - floor, 2460
  - Math, 2457
  - max, 2460–2462
  - min, 2462–2464
  - PI, 2472
  - pow, 2464
  - random, 2465
  - round, 2465, 2466
  - signum, 2466, 2467
  - sqrt, 2468
  - toDegrees, 2471
  - toRadians, 2471
- decaf::lang::Number, 2786
  - ~Number, 2787
  - byteValue, 2787
  - doubleValue, 2787
  - floatValue, 2787
  - intValue, 2788
  - longValue, 2788
  - shortValue, 2788
- decaf::lang::Pointer, 2896
  - ~Pointer, 2900
  - CounterType, 2898
  - dynamicCast, 2900
  - get, 2900
  - operator\*, 2900, 2901
  - operator->, 2901
  - operator=, 2901, 2902
  - operator==, 2902, 2903
  - Pointer, 2898, 2899
  - PointerType, 2898
  - ReferenceType, 2898
  - release, 2902
  - reset, 2902
  - staticCast, 2902
  - swap, 2902
- decaf::lang::PointerComparator, 2903
  - compare, 2904
  - operator(), 2904
- decaf::lang::Readable, 3106
  - ~Readable, 3106
  - read, 3107
- decaf::lang::Runnable, 3264
  - ~Runnable, 3265
  - run, 3265
- decaf::lang::Runtime, 3265
  - ~Runtime, 3266
  - decaf::lang::System, 3678
  - decaf::lang::Thread, 3716
  - decaf::util::logging::LogManager, 2370
  - getRuntime, 3266

- initializeRuntime, 3266
- shutdownRuntime, 3266
- decaf::lang::Short, 3380
  - ~Short, 3382
  - byteValue, 3382
  - compareTo, 3383
  - decode, 3383
  - doubleValue, 3384
  - equals, 3384
  - floatValue, 3384
  - intValue, 3384
  - longValue, 3385
  - MAX\_VALUE, 3389
  - MIN\_VALUE, 3389
  - operator<, 3385
  - operator==, 3386
  - parseShort, 3386, 3387
  - reverseBytes, 3387
  - Short, 3382
  - shortValue, 3388
  - SIZE, 3389
  - toString, 3388
  - valueOf, 3388, 3389
- decaf::lang::STATIC\_CAST\_TOKEN, 3528
- decaf::lang::String, 3610
  - ~String, 3611
  - charAt, 3611
  - isEmpty, 3612
  - length, 3612
  - String, 3611
  - subSequence, 3612
  - toString, 3612
- decaf::lang::System, 3670
  - ~System, 3672
  - arraycopy, 3672, 3673
  - availableProcessors, 3673
  - clearProperty, 3674
  - currentTimeMillis, 3674
  - decaf::lang::Runtime, 3678
  - getenv, 3674, 3675
  - getProperties, 3675
  - getProperty, 3675, 3676
  - nanoTime, 3676
  - setenv, 3677
  - setProperty, 3677
  - System, 3672
  - unsetenv, 3677
- decaf::lang::Thread, 3707
  - ~Thread, 3711
  - BLOCKED, 3710
  - currentThread, 3711
  - decaf::lang::Runtime, 3716
  - decaf::util::concurrent::locks::LockSupport, 3716
  - getId, 3711
  - getName, 3711
  - getPriority, 3711
  - getState, 3712
  - getUncaughtExceptionHandler, 3712
  - isAlive, 3712
  - join, 3712, 3713
  - MAX\_PRIORITY, 3716
  - MIN\_PRIORITY, 3716
  - NEW, 3710
  - NORM\_PRIORITY, 3716
  - run, 3713
  - RUNNABLE, 3710
  - setName, 3713
  - setPriority, 3713
  - setUncaughtExceptionHandler, 3714
  - sleep, 3714
  - SLEEPING, 3710
  - start, 3715
  - State, 3709
  - TERMINATED, 3710
  - Thread, 3710, 3711
  - TIMED\_WAITING, 3710
  - toString, 3715
  - WAITING, 3710
  - yield, 3715
- decaf::lang::Thread::UncaughtExceptionHandler, 3841
  - ~UncaughtExceptionHandler, 3841
  - uncaughtException, 3841
- decaf::lang::ThreadGroup, 3717
  - ~ThreadGroup, 3718
  - ThreadGroup, 3718
- decaf::lang::Throwable, 3724
  - ~Throwable, 3725
  - clone, 3725
  - getCause, 3726
  - getMessage, 3726
  - getStackTrace, 3726
  - getStackTraceString, 3727
  - initCause, 3727
  - printStackTrace, 3727
  - setMark, 3727
  - Throwable, 3725
- decaf::net, 134
- decaf::net::BindException, 797

- ~BindException, 799
- BindException, 798, 799
- clone, 800
- decaf::net::ConnectException, 1230
  - ~ConnectException, 1232
  - clone, 1232
  - ConnectException, 1230, 1231
- decaf::net::HttpRetryException, 1948
  - ~HttpRetryException, 1950
  - clone, 1950
  - HttpRetryException, 1949, 1950
- decaf::net::Inet4Address, 1970
  - ~Inet4Address, 1971
  - Inet4Address, 1971
  - InetAddress, 1973
  - isAnyLocalAddress, 1971
  - isLinkLocalAddress, 1971
  - isLoopbackAddress, 1972
  - isMCGlobal, 1972
  - isMCLinkLocal, 1972
  - isMCNodeLocal, 1972
  - isMCOrgLocal, 1972
  - isMCSiteLocal, 1972
  - isMulticastAddress, 1973
  - isSiteLocalAddress, 1973
- decaf::net::Inet6Address, 1973
  - ~Inet6Address, 1974
  - Inet6Address, 1974
  - InetAddress, 1974
- decaf::net::InetAddress, 1974
  - ~InetAddress, 1976
  - addressBytes, 1981
  - anyBytes, 1981
  - bytesToInt, 1977
  - getAddress, 1977
  - getAnyAddress, 1977
  - getByAddress, 1977, 1978
  - getHostAddress, 1978
  - getHostName, 1978
  - getLocalHost, 1978
  - getLoopbackAddress, 1979
  - hostname, 1982
  - InetAddress, 1976
  - isAnyLocalAddress, 1979
  - isLinkLocalAddress, 1979
  - isLoopbackAddress, 1979
  - isMCGlobal, 1980
  - isMCLinkLocal, 1980
  - isMCNodeLocal, 1980
  - isMCOrgLocal, 1980
  - isMCSiteLocal, 1980
  - isMulticastAddress, 1981
  - isSiteLocalAddress, 1981
  - loopbackBytes, 1982
  - reached, 1982
  - toString, 1981
- decaf::net::InetSocketAddress, 1982
  - ~InetSocketAddress, 1982
  - InetSocketAddress, 1982
- decaf::net::MalformedURLException, 2416
  - ~MalformedURLException, 2418
  - clone, 2418
  - MalformedURLException, 2417, 2418
- decaf::net::NoRouteToHostException, 2773
  - ~NoRouteToHostException, 2775
  - clone, 2775
  - NoRouteToHostException, 2774, 2775
- decaf::net::PortUnreachableException, 2922
  - ~PortUnreachableException, 2924
  - clone, 2924
  - PortUnreachableException, 2923, 2924
- decaf::net::ProtocolException, 3083
  - ~ProtocolException, 3085
  - clone, 3085
  - ProtocolException, 3084, 3085
- decaf::net::ServerSocket, 3292
  - ~ServerSocket, 3295
  - accept, 3296
  - bind, 3296, 3297
  - checkClosed, 3297
  - close, 3297
  - ensureCreated, 3297
  - getDefaultBacklog, 3298
  - getLocalPort, 3298
  - getReceiveBufferSize, 3298
  - getReuseAddress, 3298
  - getSoTimeout, 3299
  - implAccept, 3299
  - isBound, 3299
  - isClosed, 3299
  - ServerSocket, 3294, 3295
  - setReceiveBufferSize, 3299
  - setReuseAddress, 3300
  - setSocketImplFactory, 3300
  - setSoTimeout, 3300
  - setupSocketImpl, 3301
  - toString, 3301
- decaf::net::ServerSocketFactory, 3301
  - ~ServerSocketFactory, 3302
  - createServerSocket, 3302–3304



- getDefault, 3304
- ServerSocketFactory, 3302
- decaf::net::Socket, 3445
  - ~Socket, 3451
  - accepted, 3451
  - bind, 3451
  - checkClosed, 3452
  - close, 3452
  - connect, 3452, 3453
  - ensureCreated, 3453
  - getInetAddress, 3453
  - getInputStream, 3453
  - getKeepAlive, 3454
  - getLocalAddress, 3454
  - getLocalPort, 3454
  - getOOBInline, 3454
  - getOutputStream, 3455
  - getPort, 3455
  - getReceiveBufferSize, 3455
  - getReuseAddress, 3456
  - getSendBufferSize, 3456
  - getSoLinger, 3456
  - getSoTimeout, 3456
  - getTcpNoDelay, 3457
  - getTrafficClass, 3457
  - impl, 3463
  - initSocketImpl, 3457
  - isBound, 3458
  - isClosed, 3458
  - isConnected, 3458
  - isInputShutdown, 3458
  - isOutputShutdown, 3458
  - sendUrgentData, 3458
  - ServerSocket, 3463
  - setKeepAlive, 3459
  - setOOBInline, 3459
  - setReceiveBufferSize, 3459
  - setReuseAddress, 3460
  - setSendBufferSize, 3460
  - setSocketImplFactory, 3460
  - setSoLinger, 3461
  - setSoTimeout, 3461
  - setTcpNoDelay, 3461
  - setTrafficClass, 3462
  - shutdownInput, 3462
  - shutdownOutput, 3462
  - Socket, 3449–3451
  - toString, 3463
- decaf::net::SocketAddress, 3463
  - ~SocketAddress, 3464
- decaf::net::SocketError, 3464
  - getErrorCode, 3464
  - getErrorString, 3464
- decaf::net::SocketException, 3465
  - ~SocketException, 3466
  - clone, 3467
  - SocketException, 3465, 3466
- decaf::net::SocketFactory, 3467
  - ~SocketFactory, 3468
  - createSocket, 3468–3470
  - getDefault, 3471
  - SocketFactory, 3468
- decaf::net::SocketImpl, 3472
  - ~SocketImpl, 3474
  - accept, 3474
  - address, 3480
  - available, 3475
  - bind, 3475
  - close, 3475
  - connect, 3476
  - create, 3476
  - fd, 3481
  - getFileDescriptor, 3476
  - getInetAddress, 3477
  - getInputStream, 3477
  - getLocalAddress, 3477
  - getLocalPort, 3477
  - getOption, 3478
  - getOutputStream, 3478
  - getPort, 3478
  - listen, 3478
  - localPort, 3481
  - port, 3481
  - sendUrgentData, 3479
  - setOption, 3479
  - shutdownInput, 3479
  - shutdownOutput, 3480
  - SocketImpl, 3474
  - supportsUrgentData, 3480
  - toString, 3480
- decaf::net::SocketImplFactory, 3481
  - ~SocketImplFactory, 3482
  - createSocketImpl, 3482
- decaf::net::SocketOptions, 3482
  - ~SocketOptions, 3483
  - SOCKET\_OPTION\_BINDADDR, 3483
  - SOCKET\_OPTION\_BROADCAST, 3484
  - SOCKET\_OPTION\_IP\_MULTICAST\_  
IF, 3484

- SOCKET\_OPTION\_IP\_MULTICAST\_ -  
IF2, 3484
- SOCKET\_OPTION\_IP\_MULTICAST\_ -  
LOOP, 3484
- SOCKET\_OPTION\_IP\_TOS, 3484
- SOCKET\_OPTION\_KEEPALIVE, 3485
- SOCKET\_OPTION\_LINGER, 3485
- SOCKET\_OPTION\_OOINLINE, 3485
- SOCKET\_OPTION\_RCVBUF, 3485
- SOCKET\_OPTION\_REUSEADDR, 3486
- SOCKET\_OPTION\_SNDBUF, 3486
- SOCKET\_OPTION\_TCP\_NODELAY,  
3486
- SOCKET\_OPTION\_TIMEOUT, 3486
- decaf::net::SocketTimeoutException, 3487
- ~SocketTimeoutException, 3489
- clone, 3489
- SocketTimeoutException, 3487, 3488
- decaf::net::ssl, 136
- decaf::net::ssl::SSLContext, 3489
- ~SSLContext, 3490
- getDefault, 3490
- getDefaultSSLParameters, 3490
- getServerSocketFactory, 3491
- getSocketFactory, 3491
- getSupportedSSLParameters, 3491
- setDefault, 3491
- SSLContext, 3490
- decaf::net::ssl::SSLContextSpi, 3492
- ~SSLContextSpi, 3493
- providerGetDefaultSSLParameters, 3493
- providerGetServerSocketFactory, 3493
- providerGetSocketFactory, 3493
- providerGetSupportedSSLParameters,  
3494
- providerInit, 3494
- decaf::net::ssl::SSLParameters, 3495
- ~SSLParameters, 3496
- getCipherSuites, 3496
- getNeedClientAuth, 3496
- getProtocols, 3497
- getWantClientAuth, 3497
- setCipherSuites, 3497
- setNeedClientAuth, 3497
- setProtocols, 3497
- setWantClientAuth, 3498
- SSLParameters, 3496
- decaf::net::ssl::SSLServerSocket, 3498
- ~SSLServerSocket, 3501
- getEnabledCipherSuites, 3501
- getEnabledProtocols, 3501
- getNeedClientAuth, 3502
- getSupportedCipherSuites, 3502
- getSupportedProtocols, 3502
- getWantClientAuth, 3502
- setEnabledCipherSuites, 3502
- setEnabledProtocols, 3503
- setNeedClientAuth, 3503
- setWantClientAuth, 3504
- SSLServerSocket, 3499, 3500
- decaf::net::ssl::SSLServerSocketFactory, 3504
- ~SSLServerSocketFactory, 3505
- getDefault, 3505
- getDefaultCipherSuites, 3505
- getSupportedCipherSuites, 3506
- SSLServerSocketFactory, 3505
- decaf::net::ssl::SSLSocket, 3506
- ~SSLSocket, 3510
- getEnabledCipherSuites, 3510
- getEnabledProtocols, 3510
- getNeedClientAuth, 3510
- getSSLParameters, 3511
- getSupportedCipherSuites, 3511
- getSupportedProtocols, 3511
- getUseClientMode, 3511
- getWantClientAuth, 3512
- setEnabledCipherSuites, 3512
- setEnabledProtocols, 3512
- setNeedClientAuth, 3513
- setSSLParameters, 3513
- setUseClientMode, 3514
- setWantClientAuth, 3514
- SSLSocket, 3508, 3509
- startHandshake, 3514
- decaf::net::ssl::SSLSocketFactory, 3515
- ~SSLSocketFactory, 3516
- createSocket, 3516
- getDefault, 3517
- getDefaultCipherSuites, 3517
- getSupportedCipherSuites, 3517
- SSLSocketFactory, 3516
- decaf::net::UnknownHostException, 3841
- ~UnknownHostException, 3843
- clone, 3844
- UnknownHostException, 3842, 3843
- decaf::net::UnknownServiceException, 3844
- ~UnknownServiceException, 3846
- clone, 3846
- UnknownServiceException, 3845, 3846
- decaf::net::URI, 3853

- ~URI, 3857
- compareTo, 3857
- create, 3857
- equals, 3858
- getAuthority, 3858
- getFragment, 3858
- getHost, 3858
- getPath, 3858
- getPort, 3858
- getQuery, 3858
- getRawAuthority, 3858
- getRawFragment, 3859
- getRawPath, 3859
- getRawQuery, 3859
- getRawSchemeSpecificPart, 3859
- getRawUserInfo, 3860
- getScheme, 3860
- getSchemeSpecificPart, 3860
- getUserInfo, 3860
- isAbsolute, 3860
- isOpaque, 3861
- normalize, 3861
- operator<, 3861
- operator==, 3862
- parseServerAuthority, 3862
- relativize, 3862
- resolve, 3863
- toString, 3864
- toURL, 3864
- URI, 3855–3857
- decaf::net::URISyntaxException, 3880
  - ~URISyntaxException, 3883
  - clone, 3883
  - getIndex, 3883
  - getInput, 3883
  - getReason, 3883
  - URISyntaxException, 3881, 3882
- decaf::net::URL, 3891
  - ~URL, 3893
  - URL, 3893
- decaf::net::URLDecoder, 3893
  - ~URLDecoder, 3894
  - decode, 3894
- decaf::net::URLEncoder, 3894
  - ~URLEncoder, 3894
  - encode, 3895
- decaf::nio, 136
- decaf::nio::Buffer, 887
  - ~Buffer, 889
  - \_capacity, 893
  - \_limit, 893
  - \_mark, 893
  - \_markSet, 893
  - \_position, 893
  - Buffer, 889
  - capacity, 889
  - clear, 890
  - flip, 890
  - hasRemaining, 890
  - isReadOnly, 890
  - limit, 891
  - mark, 891
  - position, 891, 892
  - remaining, 892
  - reset, 892
  - rewind, 892
- decaf::nio::BufferOverflowException, 914
  - ~BufferOverflowException, 915
  - BufferOverflowException, 914, 915
  - clone, 916
- decaf::nio::BufferUnderflowException, 916
  - ~BufferUnderflowException, 918
  - BufferUnderflowException, 917, 918
  - clone, 918
- decaf::nio::ByteBuffer, 995
  - ~ByteBuffer, 1000
  - allocate, 1000
  - array, 1001
  - arrayOffset, 1001
  - asCharBuffer, 1002
  - asDoubleBuffer, 1002
  - asFloatBuffer, 1002
  - asIntBuffer, 1003
  - asLongBuffer, 1003
  - asReadOnlyBuffer, 1003
  - asShortBuffer, 1004
  - ByteBuffer, 1000
  - compact, 1004
  - compareTo, 1005
  - duplicate, 1005
  - equals, 1005
  - get, 1005, 1006
  - getChar, 1007
  - getDouble, 1008
  - getFloat, 1009
  - getInt, 1009, 1010
  - getLong, 1010
  - getShort, 1011
  - hasArray, 1012
  - isReadOnly, 1012

- operator<, 1012
- operator==, 1012
- put, 1012–1015
- putChar, 1015, 1016
- putDouble, 1016, 1017
- putFloat, 1017, 1018
- putInt, 1018, 1019
- putLong, 1019, 1020
- putShort, 1020, 1021
- slice, 1021
- toString, 1022
- wrap, 1022
- decaf::nio::CharBuffer, 1089
  - ~CharBuffer, 1092
  - allocate, 1092
  - append, 1093, 1094
  - array, 1094
  - arrayOffset, 1095
  - asReadOnlyBuffer, 1095
  - charAt, 1096
  - CharBuffer, 1092
  - compact, 1096
  - compareTo, 1097
  - duplicate, 1097
  - equals, 1097
  - get, 1097, 1098
  - hasArray, 1099
  - length, 1099
  - operator<, 1100
  - operator==, 1100
  - put, 1100–1104
  - read, 1104
  - slice, 1105
  - subSequence, 1105
  - toString, 1106
  - wrap, 1106
- decaf::nio::DoubleBuffer, 1773
  - ~DoubleBuffer, 1776
  - allocate, 1776
  - array, 1776
  - arrayOffset, 1777
  - asReadOnlyBuffer, 1777
  - compact, 1778
  - compareTo, 1778
  - DoubleBuffer, 1776
  - duplicate, 1778
  - equals, 1779
  - get, 1779, 1780
  - hasArray, 1781
  - operator<, 1781
  - operator==, 1781
  - put, 1781–1783
  - slice, 1784
  - toString, 1784
  - wrap, 1785
- decaf::nio::FloatBuffer, 1887
  - ~FloatBuffer, 1890
  - allocate, 1890
  - array, 1890
  - arrayOffset, 1891
  - asReadOnlyBuffer, 1891
  - compact, 1891
  - compareTo, 1892
  - duplicate, 1892
  - equals, 1892
  - FloatBuffer, 1889
  - get, 1892–1894
  - hasArray, 1894
  - operator<, 1895
  - operator==, 1895
  - put, 1895–1897
  - slice, 1898
  - toString, 1898
  - wrap, 1898, 1899
- decaf::nio::IntBuffer, 2026
  - ~IntBuffer, 2029
  - allocate, 2029
  - array, 2029
  - arrayOffset, 2029
  - asReadOnlyBuffer, 2030
  - compact, 2030
  - compareTo, 2031
  - duplicate, 2031
  - equals, 2031
  - get, 2031–2033
  - hasArray, 2033
  - IntBuffer, 2028
  - operator<, 2034
  - operator==, 2034
  - put, 2034–2036
  - slice, 2037
  - toString, 2037
  - wrap, 2037, 2038
- decaf::nio::InvalidMarkException, 2096
  - ~InvalidMarkException, 2098
  - clone, 2098
  - InvalidMarkException, 2097, 2098
- decaf::nio::LongBuffer, 2403
  - ~LongBuffer, 2406
  - allocate, 2406

- array, 2406
- arrayOffset, 2406
- asReadOnlyBuffer, 2407
- compact, 2407
- compareTo, 2408
- duplicate, 2408
- equals, 2408
- get, 2408–2410
- hasArray, 2410
- LongBuffer, 2405
- operator<, 2411
- operator==, 2411
- put, 2411–2413
- slice, 2414
- toString, 2414
- wrap, 2414, 2415
- decaf::nio::ReadOnlyBufferException, 3115
  - ~ReadOnlyBufferException, 3116
  - clone, 3117
  - ReadOnlyBufferException, 3115, 3116
- decaf::nio::ShortBuffer, 3401
  - ~ShortBuffer, 3403
  - allocate, 3403
  - array, 3404
  - arrayOffset, 3404
  - asReadOnlyBuffer, 3405
  - compact, 3405
  - compareTo, 3405
  - duplicate, 3406
  - equals, 3406
  - get, 3406–3408
  - hasArray, 3408
  - operator<, 3408
  - operator==, 3408
  - put, 3408–3410
  - ShortBuffer, 3403
  - slice, 3411
  - toString, 3411
  - wrap, 3412
- decaf::security, 137
- decaf::security::auth, 137
- decaf::security::auth::x500, 137
- decaf::security::auth::x500::X500Principal, 3957
  - ~X500Principal, 3957
  - getEncoded, 3957
  - getName, 3957
  - hashCode, 3958
- decaf::security::cert, 138
- decaf::security::cert::Certificate, 1055
  - ~Certificate, 1056
  - equals, 1056
  - getEncoded, 1056
  - getPublicKey, 1057
  - getType, 1057
  - toString, 1057
  - verify, 1057, 1058
- decaf::security::cert::CertificateEncodingException, 1059
  - ~CertificateEncodingException, 1060
  - CertificateEncodingException, 1060
  - clone, 1060
- decaf::security::cert::CertificateException, 1061
  - ~CertificateException, 1062
  - CertificateException, 1061, 1062
  - clone, 1062
- decaf::security::cert::CertificateExpiredException, 1063
  - ~CertificateExpiredException, 1064
  - CertificateExpiredException, 1063, 1064
  - clone, 1064
- decaf::security::cert::CertificateNotYetValidException, 1065
  - ~CertificateNotYetValidException, 1066
  - CertificateNotYetValidException, 1065, 1066
  - clone, 1066
- decaf::security::cert::CertificateParsingException, 1067
  - ~CertificateParsingException, 1068
  - CertificateParsingException, 1067, 1068
  - clone, 1068
- decaf::security::cert::X509Certificate, 3958
  - ~X509Certificate, 3959
  - checkValidity, 3959
  - getBasicConstraints, 3959
  - getIssuerUniqueID, 3959
  - getIssuerX500Principal, 3959
  - getKeyUsage, 3959
  - getNotAfter, 3959
  - getNotBefore, 3959
  - getSigAlgName, 3959
  - getSigAlgOID, 3959
  - getSigAlgParams, 3959
  - getSignature, 3959
  - getSubjectUniqueID, 3960
  - getSubjectX500Principal, 3960
  - getTBSCertificate, 3960
  - getVersion, 3960

- decaf::security::GeneralSecurityException, 1934
  - ~GeneralSecurityException, 1936
  - clone, 1936
  - GeneralSecurityException, 1935, 1936
- decaf::security::InvalidKeyException, 2094
  - ~InvalidKeyException, 2096
  - clone, 2096
  - InvalidKeyException, 2094, 2095
- decaf::security::Key, 2253
  - ~Key, 2254
  - getAlgorithm, 2254
  - getEncoded, 2254
  - getFormat, 2254
- decaf::security::KeyException, 2255
  - ~KeyException, 2257
  - clone, 2257
  - KeyException, 2255, 2256
- decaf::security::KeyManagementException, 2257
  - ~KeyManagementException, 2259
  - clone, 2259
  - KeyManagementException, 2258, 2259
- decaf::security::NoSuchAlgorithmExceptionException, 2776
  - ~NoSuchAlgorithmExceptionException, 2778
  - clone, 2778
  - NoSuchAlgorithmExceptionException, 2776, 2777
- decaf::security::NoSuchProviderException, 2781
  - ~NoSuchProviderException, 2783
  - clone, 2783
  - NoSuchProviderException, 2782, 2783
- decaf::security::Principal, 2974
  - ~Principal, 2975
  - equals, 2975
  - getName, 2975
- decaf::security::PublicKey, 3086
  - ~PublicKey, 3086
- decaf::security::SecureRandom, 3269
  - ~SecureRandom, 3272
  - next, 3272
  - nextBytes, 3273
  - SecureRandom, 3271, 3272
  - setSeed, 3274
- decaf::security::SecureRandomSpi, 3278
  - ~SecureRandomSpi, 3279
  - providerGenerateSeed, 3279
  - providerNextBytes, 3279
  - providerSetSeed, 3279
- SecureRandomSpi, 3279
- decaf::security::SignatureException, 3440
  - ~SignatureException, 3442
  - clone, 3442
  - SignatureException, 3441, 3442
- decaf::util, 138
- decaf::util::AbstractCollection, 147
  - ~AbstractCollection, 149
  - AbstractCollection, 149
  - add, 149
  - addAll, 150
  - clear, 151
  - contains, 152
  - containsAll, 152
  - copy, 153
  - equals, 153
  - isEmpty, 154
  - lock, 154
  - mutex, 160
  - notify, 154
  - notifyAll, 155
  - operator=, 155
  - remove, 155
  - removeAll, 156
  - retainAll, 157
  - toArray, 158
  - tryLock, 158
  - unlock, 159
  - wait, 159, 160
- decaf::util::AbstractList, 161
  - ~AbstractList, 162
- decaf::util::AbstractMap, 162
  - ~AbstractMap, 163
- decaf::util::AbstractQueue, 163
  - ~AbstractQueue, 164
  - AbstractQueue, 164
  - add, 164
  - addAll, 165
  - clear, 166
  - element, 166
  - remove, 166
- decaf::util::AbstractSequentialList, 167
  - ~AbstractSequentialList, 168
- decaf::util::AbstractSet, 168
  - ~AbstractSet, 169
  - removeAll, 169
- decaf::util::Collection, 1155
  - ~Collection, 1156
  - add, 1156
  - addAll, 1157

- clear, 1158
- contains, 1159
- containsAll, 1160
- equals, 1160
- isEmpty, 1161
- remove, 1162
- removeAll, 1162
- retainAll, 1163
- size, 1164
- toArray, 1165
- decaf::util::Comparator, 1189
  - ~Comparator, 1190
  - compare, 1190
  - operator(), 1190
- decaf::util::comparators, 140
- decaf::util::comparators::Less, 2287
  - ~Less, 2287
  - compare, 2287
  - Less, 2287
  - operator(), 2288
- decaf::util::concurrent, 140
- decaf::util::concurrent::atomic, 141
- decaf::util::concurrent::atomic::AtomicBoolean, 705
  - ~AtomicBoolean, 706
  - AtomicBoolean, 706
  - compareAndSet, 706
  - get, 707
  - getAndSet, 707
  - set, 707
  - toString, 707
- decaf::util::concurrent::atomic::AtomicInteger, 708
  - ~AtomicInteger, 709
  - addAndGet, 709
  - AtomicInteger, 709
  - compareAndSet, 710
  - decrementAndGet, 710
  - doubleValue, 710
  - floatValue, 710
  - get, 711
  - getAndAdd, 711
  - getAndDecrement, 711
  - getAndIncrement, 711
  - getAndSet, 711
  - incrementAndGet, 712
  - intValue, 712
  - longValue, 712
  - set, 712
  - toString, 713
- decaf::util::concurrent::atomic::AtomicReferenceCounter, 713
  - ~AtomicReferenceCounter, 714
  - AtomicReferenceCounter, 714
  - release, 714
  - swap, 715
- decaf::util::concurrent::atomic::AtomicReference, 716
  - ~AtomicReference, 716
  - AtomicReference, 716
  - compareAndSet, 717
  - get, 717
  - getAndSet, 717
  - set, 717
  - toString, 718
- decaf::util::concurrent::BlockingQueue, 804
  - ~BlockingQueue, 807
  - drainTo, 807
  - offer, 808
  - poll, 809
  - put, 809
  - remainingCapacity, 810
  - take, 810
- decaf::util::concurrent::BrokenBarrierException, 820
  - ~BrokenBarrierException, 822
  - BrokenBarrierException, 821, 822
  - clone, 822
- decaf::util::concurrent::Callable, 1051
  - ~Callable, 1052
  - call, 1052
- decaf::util::concurrent::CancellationException, 1052
  - ~CancellationException, 1054
  - CancellationException, 1053, 1054
  - clone, 1055
- decaf::util::concurrent::ConcurrentMap, 1198
  - ~ConcurrentMap, 1199
  - putIfAbsent, 1199
  - remove, 1200
  - replace, 1201
- decaf::util::concurrent::ConcurrentStlMap, 1203
  - ~ConcurrentStlMap, 1207
  - clear, 1207
  - ConcurrentStlMap, 1207
  - containsKey, 1208
  - containsValue, 1208
  - copy, 1209
  - equals, 1209
  - get, 1210

- isEmpty, 1211
- keySet, 1211
- lock, 1211
- notify, 1211
- notifyAll, 1212
- put, 1212
- putAll, 1213
- putIfAbsent, 1213
- remove, 1214, 1215
- replace, 1215, 1216
- size, 1217
- tryLock, 1217
- unlock, 1217
- values, 1217
- wait, 1218, 1219
- decaf::util::concurrent::ConditionHandle, 1226
  - ~ConditionHandle, 1227
  - condition, 1227
  - ConditionHandle, 1227
  - criticalSection, 1227
  - generation, 1227
  - mutex, 1227
  - numWaiting, 1227
  - numWake, 1227
  - semaphore, 1227
- decaf::util::concurrent::CountDownLatch, 1487
  - ~CountDownLatch, 1487
  - await, 1487–1489
  - countDown, 1489
  - CountDownLatch, 1487
  - getCount, 1489
- decaf::util::concurrent::Delayed, 1686
  - ~Delayed, 1687
  - getDelay, 1687
- decaf::util::concurrent::ExecutionException, 1829
  - ~ExecutionException, 1831
  - clone, 1831
  - ExecutionException, 1829, 1830
- decaf::util::concurrent::Executor, 1831
  - ~Executor, 1833
  - execute, 1833
- decaf::util::concurrent::ExecutorService, 1833
  - ~ExecutorService, 1834
  - awaitTermination, 1834
- decaf::util::concurrent::Future, 1929
  - ~Future, 1930
  - cancel, 1930
  - get, 1931
  - isCancelled, 1932
  - isDone, 1932
- decaf::util::concurrent::Lock, 2334
  - ~Lock, 2335
  - isLocked, 2335
  - Lock, 2335
  - lock, 2335
  - unlock, 2335
- decaf::util::concurrent::locks, 142
- decaf::util::concurrent::locks::Condition, 1220
  - ~Condition, 1222
  - await, 1222, 1223
  - awaitNanos, 1223
  - awaitUninterruptibly, 1225
  - awaitUntil, 1226
  - signal, 1226
  - signalAll, 1226
- decaf::util::concurrent::locks::Lock, 2336
  - ~Lock, 2338
  - lock, 2338
  - lockInterruptibly, 2338
  - newCondition, 2339
  - tryLock, 2339, 2340
  - unlock, 2341
- decaf::util::concurrent::locks::LockSupport, 2341
  - ~LockSupport, 2343
- decaf::lang::Thread, 3716
- park, 2343
- parkNanos, 2343
- parkUntil, 2344
- unpark, 2344
- decaf::util::concurrent::locks::ReadWriteLock, 3117
  - ~ReadWriteLock, 3118
  - readLock, 3119
  - writeLock, 3119
- decaf::util::concurrent::locks::ReentrantLock, 3126
  - ~ReentrantLock, 3128
  - getHoldCount, 3128
  - isFair, 3129
  - isHeldByCurrentThread, 3129
  - isLocked, 3129
  - lock, 3129
  - lockInterruptibly, 3130
  - newCondition, 3131
  - ReentrantLock, 3128
  - toString, 3131
  - tryLock, 3131, 3132
  - unlock, 3133



- decaf::util::concurrent::Mutex, 2736
  - ~Mutex, 2737
  - lock, 2737
  - Mutex, 2737
  - notify, 2737
  - notifyAll, 2738
  - tryLock, 2738
  - unlock, 2739
  - wait, 2739, 2740
- decaf::util::concurrent::MutexHandle, 2741
  - ~MutexHandle, 2741
  - lock\_count, 2741
  - lock\_owner, 2741
  - mutex, 2741, 2742
  - MutexHandle, 2741
- decaf::util::concurrent::PooledThread, 2918
  - ~PooledThread, 2919
  - getPooledThreadListener, 2919
  - isBusy, 2919
  - PooledThread, 2919
  - run, 2919
  - setPooledThreadListener, 2920
  - stop, 2920
- decaf::util::concurrent::PooledThreadListener, 2920
  - ~PooledThreadListener, 2921
  - onTaskCompleted, 2921
  - onTaskException, 2921
  - onTaskStarted, 2921
- decaf::util::concurrent::RejectedExecutionException, 3134
  - ~RejectedExecutionException, 3136
  - clone, 3136
  - RejectedExecutionException, 3134, 3135
- decaf::util::concurrent::RejectedExecutionHandler, 3136
  - ~RejectedExecutionHandler, 3137
  - rejectedExecution, 3137
- decaf::util::concurrent::Semaphore, 3280
  - ~Semaphore, 3283
  - acquire, 3283, 3284
  - acquireUninterruptibly, 3284, 3285
  - availablePermits, 3285
  - drainPermits, 3286
  - isFair, 3286
  - release, 3286
  - Semaphore, 3283
  - toString, 3287
  - tryAcquire, 3287–3289
- decaf::util::concurrent::Synchronizable, 3644
  - ~Synchronizable, 3645
  - lock, 3645
  - notify, 3646
  - notifyAll, 3647
  - tryLock, 3648
  - unlock, 3650
  - wait, 3651–3653
- decaf::util::concurrent::SynchronousQueue, 3660
  - ~SynchronousQueue, 3662
  - clear, 3662
  - contains, 3663
  - containsAll, 3663
  - drainTo, 3663, 3664
  - equals, 3665
  - isEmpty, 3665
  - iterator, 3665
  - offer, 3665, 3666
  - peek, 3667
  - poll, 3667
  - put, 3667
  - remainingCapacity, 3668
  - remove, 3668
  - removeAll, 3668
  - retainAll, 3669
  - size, 3669
  - SynchronousQueue, 3662
  - take, 3669
  - toArray, 3669
- decaf::util::concurrent::TaskListener, 3679
  - ~TaskListener, 3679
  - onTaskComplete, 3679
  - onTaskException, 3680
- decaf::util::concurrent::ThreadFactory, 3716
  - ~ThreadFactory, 3717
  - newThread, 3717
- decaf::util::concurrent::ThreadPool, 3718
  - ~ThreadPool, 3720
  - DEFAULT\_MAX\_BLOCK\_SIZE, 3723
  - DEFAULT\_MAX\_POOL\_SIZE, 3724
  - deQueueTask, 3720
  - getBacklog, 3720
  - getBlockSize, 3720
  - getFreeThreadCount, 3721
  - getInstance, 3721
  - getMaxThreads, 3721
  - getPoolSize, 3721
  - onTaskCompleted, 3722
  - onTaskException, 3722
  - onTaskStarted, 3722

- queueTask, 3722
- reserve, 3723
- setBlockSize, 3723
- setMaxThreads, 3723
- Task, 3720
- ThreadPool, 3720
- decaf::util::concurrent::TimeoutException, 3728
  - ~TimeoutException, 3730
  - clone, 3730
  - TimeoutException, 3729, 3730
- decaf::util::concurrent::TimeUnit, 3748
  - ~TimeUnit, 3750
  - compareTo, 3750
  - convert, 3751
  - DAYS, 3757
  - equals, 3751
  - HOURS, 3757
  - MICROSECONDS, 3757
  - MILLISECONDS, 3757
  - MINUTES, 3757
  - NANOSECONDS, 3757
  - operator<, 3751
  - operator==, 3752
  - SECONDS, 3757
  - sleep, 3752
  - timedJoin, 3752
  - timedWait, 3753
  - TimeUnit, 3750
  - toDays, 3753
  - toHours, 3754
  - toMicros, 3754
  - toMillis, 3754
  - toMinutes, 3755
  - toNanos, 3755
  - toSeconds, 3756
  - toString, 3756
  - valueOf, 3756
  - values, 3757
- decaf::util::Date, 1633
  - ~Date, 1634
  - after, 1635
  - before, 1635
  - compareTo, 1635
  - Date, 1634
  - equals, 1635
  - getTime, 1635
  - operator<, 1636
  - operator=, 1636
  - operator==, 1636
  - setTime, 1636
  - toString, 1637
- decaf::util::Iterator, 2114
  - ~Iterator, 2115
  - hasNext, 2115
  - next, 2115
  - remove, 2115
- decaf::util::List, 2296
  - ~List, 2297
  - add, 2297
  - addAll, 2298
  - get, 2298
  - indexOf, 2299
  - lastIndexOf, 2300
  - List, 2297
  - listIterator, 2300, 2301
  - remove, 2302
  - set, 2302
- decaf::util::ListIterator, 2303
  - ~ListIterator, 2304
  - add, 2304
  - hasPrevious, 2305
  - nextIndex, 2305
  - previous, 2305
  - previousIndex, 2306
  - set, 2306
- decaf::util::logging, 142
  - Debug, 144
  - Error, 144
  - Fatal, 144
  - Info, 144
  - Levels, 143
  - Markblock, 143
  - Null, 143
  - Off, 143
  - Throwing, 144
  - Warn, 144
- decaf::util::logging::ConsoleHandler, 1367
  - ~ConsoleHandler, 1368
  - close, 1368
  - ConsoleHandler, 1368
  - publish, 1368
- decaf::util::logging::ErrorManager, 1792
  - ~ErrorManager, 1793
  - CLOSE\_FAILURE, 1793
  - error, 1793
  - ErrorManager, 1793
  - FLUSH\_FAILURE, 1793
  - FORMAT\_FAILURE, 1793
  - GENERIC\_FAILURE, 1793
  - OPEN\_FAILURE, 1793

- WRITE\_FAILURE, 1794
- decaf::util::logging::Filter, 1853
  - ~Filter, 1853
  - isLoggable, 1853
- decaf::util::logging::Formatter, 1927
  - ~Formatter, 1928
  - format, 1928
  - formatMessage, 1928
  - getHead, 1928
  - getTail, 1929
- decaf::util::logging::Handler, 1941
  - ~Handler, 1942
  - flush, 1942
  - getErrorManager, 1942
  - getFilter, 1943
  - getFormatter, 1943
  - getLevel, 1943
  - Handler, 1942
  - isLoggable, 1943
  - publish, 1943
  - reportError, 1944
  - setErrorManager, 1944
  - setFilter, 1944
  - setFormatter, 1944
  - setLevel, 1945
- decaf::util::logging::Level, 2290
  - ~Level, 2292
  - ALL, 2294
  - compareTo, 2292
  - CONFIG, 2294
  - DEBUG, 2294
  - equals, 2292
  - FINE, 2294
  - FINER, 2294
  - FINEST, 2294
  - getName, 2292
  - INFO, 2295
  - INHERIT, 2295
  - intValue, 2293
  - Level, 2292
  - OFF, 2295
  - operator<, 2293
  - operator==, 2293
  - parse, 2293
  - SEVERE, 2295
  - toString, 2293
  - WARNING, 2295
- decaf::util::logging::Logger, 2345
  - ~Logger, 2348
  - addHandler, 2348
  - config, 2349
  - debug, 2349
  - entering, 2349
  - exiting, 2350
  - fine, 2350
  - finer, 2350
  - finest, 2351
  - getAnonymousLogger, 2351
  - getFilter, 2351
  - getHandlers, 2351
  - getLevel, 2352
  - getLogger, 2352
  - getName, 2352
  - getParent, 2352
  - getUseParentHandlers, 2353
  - info, 2353
  - isLoggable, 2353
  - log, 2353, 2354
  - Logger, 2348
  - removeHandler, 2355
  - setFilter, 2355
  - setLevel, 2355
  - setParent, 2355
  - setUseParentHandlers, 2356
  - severe, 2356
  - throwing, 2356
  - warning, 2357
- decaf::util::logging::LoggerHierarchy, 2357
  - ~LoggerHierarchy, 2357
  - LoggerHierarchy, 2357
- decaf::util::logging::LogManager, 2363
  - ~LogManager, 2366
  - addLogger, 2366
  - addPropertyChangeListener, 2367
  - decaf::lang::Runtime, 2370
  - getLogger, 2367
  - getLoggerNames, 2367
  - getLogManager, 2367
  - getProperties, 2368
  - getProperty, 2368
  - LogManager, 2366
  - operator=, 2368
  - readConfiguration, 2368, 2369
  - removePropertyChangeListener, 2369
  - reset, 2369
  - setProperties, 2369
- decaf::util::logging::LogRecord, 2370
  - ~LogRecord, 2371
  - getLevel, 2371
  - getLoggerName, 2372

- getMessage, 2372
- getSourceFile, 2372
- getSourceFunction, 2372
- getSourceLine, 2372
- getThrown, 2372
- getTimestamp, 2373
- getTreadId, 2373
- LogRecord, 2371
- setLevel, 2373
- setLoggerName, 2373
- setMessage, 2373
- setSourceFile, 2374
- setSourceFunction, 2374
- setSourceLine, 2374
- setThrown, 2374
- setTimestamp, 2375
- setTreadId, 2375
- decaf::util::logging::LogWriter, 2375
  - ~LogWriter, 2376
  - destroy, 2376
  - getInstance, 2376
  - log, 2376
  - LogWriter, 2376
  - returnInstance, 2376
- decaf::util::logging::MarkBlockLogger, 2443
  - ~MarkBlockLogger, 2444
  - MarkBlockLogger, 2443
- decaf::util::logging::PropertiesChangeListener, 3082
  - ~PropertiesChangeListener, 3083
  - onPropertiesReset, 3083
  - onPropertyChanged, 3083
- decaf::util::logging::SimpleFormatter, 3442
  - ~SimpleFormatter, 3443
  - format, 3443
  - SimpleFormatter, 3443
- decaf::util::logging::SimpleLogger, 3444
  - ~SimpleLogger, 3444
  - debug, 3445
  - error, 3445
  - fatal, 3445
  - info, 3445
  - log, 3445
  - mark, 3445
  - SimpleLogger, 3444
  - warn, 3445
- decaf::util::logging::StreamHandler, 3591
  - ~StreamHandler, 3593
  - close, 3593
  - flush, 3593
  - isLoggable, 3593
  - publish, 3594
  - setOutputStream, 3594
  - StreamHandler, 3593
- decaf::util::logging::XMLFormatter, 3988
  - ~XMLFormatter, 3989
  - format, 3989
  - getHead, 3989
  - getTail, 3990
  - XMLFormatter, 3989
- decaf::util::Map, 2419
  - ~Map, 2420
  - clear, 2420
  - containsKey, 2421
  - containsValue, 2422
  - copy, 2423
  - equals, 2423
  - get, 2423, 2424
  - isEmpty, 2425
  - keySet, 2426
  - Map, 2420
  - put, 2427
  - putAll, 2428
  - remove, 2429
  - size, 2430
  - values, 2430
- decaf::util::Map::Entry, 1788
  - ~Entry, 1789
  - Entry, 1789
  - getKey, 1789
  - getValue, 1789
  - setValue, 1789
- decaf::util::PriorityQueue, 2975
  - ~PriorityQueue, 2979
  - add, 2979
  - clear, 2980
  - comparator, 2980
  - iterator, 2980
  - offer, 2980
  - operator=, 2981
  - peek, 2981
  - poll, 2982
  - PriorityQueue, 2978, 2979
  - PriorityQueueIterator, 2984
  - remove, 2982, 2983
  - size, 2983
- decaf::util::Properties, 3072
  - ~Properties, 3074
  - clear, 3074
  - clone, 3074

- copy, 3075
- defaults, 3082
- equals, 3075
- getProperty, 3075
- hasProperty, 3076
- isEmpty, 3076
- load, 3076, 3077
- operator=, 3079
- Properties, 3074
- propertyNames, 3079
- remove, 3079
- setProperty, 3079
- size, 3080
- store, 3080, 3081
- toArray, 3081
- toString, 3081
- decaf::util::Queue, 3094
  - ~Queue, 3095
  - element, 3096
  - offer, 3096
  - peek, 3097
  - poll, 3097
  - remove, 3097
- decaf::util::Random, 3100
  - next, 3102
  - nextBoolean, 3103
  - nextBytes, 3103
  - nextDouble, 3103
  - nextFloat, 3104
  - nextGaussian, 3104
  - nextInt, 3104, 3105
  - nextLong, 3105
  - Random, 3102
  - setSeed, 3105
- decaf::util::Set, 3379
  - ~Set, 3380
- decaf::util::StlList, 3529
  - ~StlList, 3535
  - add, 3535, 3536
  - addAll, 3536
  - clear, 3537
  - contains, 3537
  - copy, 3538
  - equals, 3538
  - get, 3538
  - indexOf, 3538
  - isEmpty, 3539
  - iterator, 3539
  - lastIndexOf, 3539
  - listIterator, 3540
  - remove, 3541
  - set, 3542
  - size, 3542
  - StlList, 3534
- decaf::util::StlMap, 3543
  - ~StlMap, 3547
  - clear, 3547
  - containsKey, 3548
  - containsValue, 3548
  - copy, 3548, 3549
  - equals, 3549
  - get, 3549, 3550
  - isEmpty, 3550
  - keySet, 3550
  - lock, 3551
  - notify, 3551
  - notifyAll, 3551
  - put, 3552
  - putAll, 3552
  - remove, 3553
  - size, 3553
  - StlMap, 3547
  - tryLock, 3554
  - unlock, 3554
  - values, 3554
  - wait, 3554–3556
- decaf::util::StlQueue, 3556
  - ~StlQueue, 3558
  - back, 3558, 3559
  - clear, 3559
  - empty, 3559
  - enqueueFront, 3559
  - front, 3559, 3560
  - getSafeValue, 3560
  - iterator, 3560
  - lock, 3560
  - notify, 3561
  - notifyAll, 3561
  - pop, 3561
  - push, 3561
  - reverse, 3562
  - size, 3562
  - StlQueue, 3558
  - toArray, 3562
  - tryLock, 3562
  - unlock, 3562
  - wait, 3563, 3564
- decaf::util::StlSet, 3564
  - ~StlSet, 3567
  - add, 3568

- clear, 3568
- contains, 3569
- copy, 3569
- equals, 3569
- isEmpty, 3569
- iterator, 3570
- remove, 3570
- size, 3571
- StlSet, 3567
- decaf::util::StringTokenizer, 3613
  - ~StringTokenizer, 3614
  - countTokens, 3614
  - hasMoreTokens, 3614
  - nextToken, 3614, 3615
  - reset, 3615
  - StringTokenizer, 3613
  - toArray, 3616
- decaf::util::Timer, 3730
  - ~Timer, 3733
  - cancel, 3733
  - purge, 3733
  - schedule, 3733–3738
  - scheduleAtFixedRate, 3739–3741
  - Timer, 3732
- decaf::util::TimerTask, 3743
  - ~TimerTask, 3743
  - cancel, 3744
  - decaf::internal::util::TimerTaskHeap, 3745
  - getWhen, 3744
  - isScheduled, 3744
  - scheduledExecutionTime, 3744
  - setScheduledTime, 3744
  - Timer, 3745
  - TimerImpl, 3745
  - TimerTask, 3743
- decaf::util::UUID, 3900
  - ~UUID, 3902
  - clockSequence, 3902
  - compareTo, 3903
  - equals, 3903
  - fromString, 3903
  - getLeastSignificantBits, 3903
  - getMostSignificantBits, 3903
  - nameUUIDFromBytes, 3903, 3904
  - node, 3904
  - operator<, 3904
  - operator==, 3905
  - randomUUID, 3905
  - timestamp, 3905
  - toString, 3906
  - UUID, 3902
  - variant, 3906
  - version, 3906
- decaf::util::zip, 144
- decaf::util::zip::Adler32, 691
  - ~Adler32, 692
  - Adler32, 692
  - getValue, 692
  - reset, 692
  - update, 692, 693
- decaf::util::zip::CheckedInputStream, 1109
  - ~CheckedInputStream, 1111
  - CheckedInputStream, 1110
  - doReadArrayBounded, 1111
  - doReadByte, 1111
  - getChecksum, 1111
  - skip, 1111
- decaf::util::zip::CheckedOutputStream, 1112
  - ~CheckedOutputStream, 1113
  - CheckedOutputStream, 1113
  - doWriteArrayBounded, 1113
  - doWriteByte, 1114
  - getChecksum, 1114
- decaf::util::zip::Checksum, 1114
  - ~Checksum, 1115
  - getValue, 1115
  - reset, 1115
  - update, 1115, 1116
- decaf::util::zip::CRC32, 1490
  - ~CRC32, 1491
  - CRC32, 1491
  - getValue, 1491
  - reset, 1491
  - update, 1491, 1492
- decaf::util::zip::DataFormatException, 1520
  - ~DataFormatException, 1522
  - clone, 1522
  - DataFormatException, 1521, 1522
- decaf::util::zip::Deflater, 1672
  - ~Deflater, 1674
  - BEST\_COMPRESSION, 1681
  - BEST\_SPEED, 1681
  - DEFAULT\_COMPRESSION, 1681
  - DEFAULT\_STRATEGY, 1681
  - deflate, 1674, 1675
  - DEFLATED, 1681
  - Deflater, 1674
  - end, 1676
  - FILTERED, 1681
  - finish, 1676

- finished, 1676
- getAdler, 1676
- getBytesRead, 1677
- getBytesWritten, 1677
- HUFFMAN\_ONLY, 1681
- needsInput, 1677
- NO\_COMPRESSION, 1681
- reset, 1677
- setDictionary, 1677, 1678
- setInput, 1679
- setLevel, 1680
- setStrategy, 1680
- decaf::util::zip::DeflaterOutputStream, 1682
  - ~DeflaterOutputStream, 1684
  - buf, 1686
  - close, 1685
  - DEFAULT\_BUFFER\_SIZE, 1686
  - deflate, 1685
  - deflater, 1686
  - DeflaterOutputStream, 1683, 1684
  - doWriteArrayBounded, 1685
  - doWriteByte, 1685
  - finish, 1685
  - isDone, 1686
  - ownDeflater, 1686
- decaf::util::zip::Inflater, 1985
  - ~Inflater, 1987
  - end, 1987
  - finish, 1987
  - finished, 1988
  - getAdler, 1988
  - getBytesRead, 1988
  - getBytesWritten, 1988
  - getRemaining, 1988
  - inflate, 1989, 1990
  - Inflater, 1987
  - needsDictionary, 1990
  - needsInput, 1990
  - reset, 1991
  - setDictionary, 1991, 1992
  - setInput, 1992, 1993
- decaf::util::zip::InflaterInputStream, 1994
  - ~InflaterInputStream, 1998
  - atEOF, 2001
  - available, 1998
  - buff, 2001
  - close, 1998
  - DEFAULT\_BUFFER\_SIZE, 2001
  - doReadArrayBounded, 1999
  - doReadByte, 1999
  - fill, 1999
  - inflater, 2001
  - InflaterInputStream, 1997
  - length, 2001
  - mark, 1999
  - markSupported, 2000
  - ownInflater, 2002
  - reset, 2000
  - skip, 2000
- decaf::util::zip::ZipException, 3991
  - ~ZipException, 3993
  - clone, 3993
  - ZipException, 3992, 3993
- DECAF\_API
  - decaf/util/Config.h, 4087
- DECAF\_CATCH\_EXCEPTION\_CONVERT
  - decaf/lang/exceptions/ExceptionDefines.h, 4054
- DECAF\_CATCH\_NOTHROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4054
- DECAF\_CATCH\_RETHROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4054
- DECAF\_CATCHALL\_NOTHROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4055
- DECAF\_CATCHALL\_THROW
  - decaf/lang/exceptions/ExceptionDefines.h, 4055
- DECAF\_UNUSED
  - decaf/util/Config.h, 4087
- DecafRuntime
  - decaf::internal::DecafRuntime, 1638
- decode
  - decaf::internal::net::URLEncoderDecoder, 3866
  - decaf::lang::Byte, 921
  - decaf::lang::Integer, 2042
  - decaf::lang::Long, 2381
  - decaf::lang::Short, 3383
  - decaf::net::URLDecoder, 3894
- decreaseUsage
  - activemq::util::MemoryUsage, 2473
  - activemq::util::Usage, 3896
- decrementAndGet
  - decaf::util::concurrent::atomic::AtomicInteger, 710
- DedicatedTaskRunner

activemq::threads::DedicatedTaskRunner,	activemq::core::policies::DefaultRedeliveryPolicy,
1639	1645
DEF_MEM_LEVEL	defaults
zutil.h, 4438	decaf::util::Properties, 3082
DEF_WBITS	DefaultServerSocketFactory
zutil.h, 4438	decaf::internal::net::DefaultServerSocketFactory,
DEFAULT_BUFFER_SIZE	1650
decaf::util::zip::DeflaterOutputStream,	DefaultSocketFactory
1686	decaf::internal::net::DefaultSocketFactory,
decaf::util::zip::InflaterInputStream, 2001	1654
DEFAULT_COMPRESSION	DefaultSSLContext
decaf::util::zip::Deflater, 1681	decaf::internal::net::ssl::DefaultSSLContext,
DEFAULT_DURABLE_TOPIC_PREFETCH	1657
activemq::core::policies::DefaultPrefetchPolicy,	DefaultSSLServerSocketFactory
1643	decaf::internal::net::ssl::DefaultSSLServerSocketFactory,
DEFAULT_MAX_BLOCK_SIZE	1660
decaf::util::concurrent::ThreadPool, 3723	DefaultSSLSocketFactory
DEFAULT_MAX_POOL_SIZE	decaf::internal::net::ssl::DefaultSSLSocketFactory,
decaf::util::concurrent::ThreadPool, 3724	1666
DEFAULT_MESSAGE_SIZE	deflate
activemq::commands::Message, 2491	decaf::util::zip::Deflater, 1674, 1675
DEFAULT_ORDERED_TARGET	decaf::util::zip::DeflaterOutputStream,
activemq::commands::ActiveMQDestination,	1685
303	deflate.h
DEFAULT_PRIORITY	_dist_code, 4421
activemq::cmsutil::CmsTemplate, 1153	_length_code, 4421
DEFAULT_QUEUE_BROWSER_PREFETCH	_tr_tally_dist, 4419
activemq::core::policies::DefaultPrefetchPolicy,	_tr_tally_lit, 4419
1644	BL_CODES, 4419
DEFAULT_QUEUE_PREFETCH	BUSY_STATE, 4419
activemq::core::policies::DefaultPrefetchPolicy,	Code, 4419
1644	COMMENT_STATE, 4419
DEFAULT_STRATEGY	ct_data, 4420
decaf::util::zip::Deflater, 1681	d_code, 4419
DEFAULT_TIME_TO_LIVE	D_CODES, 4420
activemq::cmsutil::CmsTemplate, 1153	Dad, 4420
DEFAULT_TOPIC_PREFETCH	deflate_state, 4420
activemq::core::policies::DefaultPrefetchPolicy,	EXTRA_STATE, 4420
1644	FINISH_STATE, 4420
DEFAULT_URI	Freq, 4420
activemq::core::ActiveMQConnectionFactory,	GZIP, 4420
275	HCRC_STATE, 4420
DEFAULT_VERSION	HEAP_SIZE, 4420
activemq::wireformat::openwire::OpenWireFormat,	INIT_STATE, 4420
2849	INITIAL_STATE, 4421
DefaultPrefetchPolicy	L_CODES, 4420
activemq::core::policies::DefaultPrefetchPolicy,	Len, 4420
1641	LENGTH_CODES, 4420
DefaultRedeliveryPolicy	LITERALS, 4420
	MAX_BITS, 4420



- MAX\_DIST, 4420
- max\_insert\_length, 4420
- MIN\_LOOKAHEAD, 4420
- NAME\_STATE, 4420
- OF, 4421
- Pos, 4421
- Posf, 4421
- put\_byte, 4420
- static\_tree\_desc, 4421
- tree\_desc, 4421
- WIN\_INIT, 4420
- deflate\_state
  - deflate.h, 4420
- DEFLATED
  - decaf::util::zip::Deflater, 1681
- deflateInit
  - zlib.h, 4433
- deflateInit2
  - zlib.h, 4433
- Deflater
  - decaf::util::zip::Deflater, 1674
- deflater
  - decaf::util::zip::DeflaterOutputStream, 1686
- DeflaterOutputStream
  - decaf::util::zip::DeflaterOutputStream, 1683, 1684
- deleteIfCancelled
  - decaf::internal::util::TimerTaskHeap, 3746
- deliverAcks
  - activemq::core::ActiveMQConsumer, 286
  - activemq::core::ActiveMQSession, 496
- DELIVERY\_MODE
  - cms::DeliveryMode, 1688
- deliverySequenced
  - activemq::commands::MessageDispatchNotification, 2594
- depth
  - internal\_state, 2082
- dequeue
  - activemq::core::ActiveMQConsumer, 286
  - activemq::core::MessageDispatchChannel, 2561
- dequeueNoWait
  - activemq::core::MessageDispatchChannel, 2561
- deQueueTask
  - decaf::util::concurrent::ThreadPool, 3720
- descriptor
  - decaf::io::FileDescriptor, 1852
- destination
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3014
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3121
  - activemq::commands::ConsumerControl, 1373
  - activemq::commands::ConsumerInfo, 1433
  - activemq::commands::DestinationInfo, 1695
  - activemq::commands::JournalQueueAck, 2119
  - activemq::commands::JournalTopicAck, 2147
  - activemq::commands::Message, 2491
  - activemq::commands::MessageAck, 2525
  - activemq::commands::MessageDispatch, 2559
  - activemq::commands::MessageDispatchNotification, 2594
  - activemq::commands::MessagePull, 2699
  - activemq::commands::ProducerInfo, 3047
  - activemq::commands::SubscriptionInfo, 3620
- DESTINATION\_ADD\_OPERATION
  - activemq::core::ActiveMQConstants, 280
- DESTINATION\_REMOVE\_OPERATION
  - activemq::core::ActiveMQConstants, 280
- DestinationActions
  - activemq::core::ActiveMQConstants, 280
- DestinationInfo
  - activemq::commands::DestinationInfo, 1693
- DestinationInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1709
  - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1697
  - activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1701
  - activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1705
  - activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1717
  - activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 1713
- DestinationOption
  - activemq::core::ActiveMQConstants, 280
- DestinationType

cms::Destination, 1689  
 destOptionMap  
   activemq::core::ActiveMQConstants::StaticInitializer, 3529  
 destOptions  
   activemq::core::ActiveMQConstants::StaticInitializer, 3529  
 destroy  
   activemq::cmsutil::CmsAccessor, 1125  
   activemq::cmsutil::CmsDestinationAccessor, 1129  
   activemq::cmsutil::CmsTemplate, 1143  
   activemq::cmsutil::DestinationResolver, 1721  
   activemq::cmsutil::DynamicDestinationResolver, 1787  
   activemq::cmsutil::ResourceLifecycleManager, 3227  
   activemq::commands::ActiveMQTempQueue, 576  
   activemq::commands::ActiveMQTempTopic, 604  
   cms::TemporaryQueue, 3702  
   cms::TemporaryTopic, 3704  
   decaf::internal::util::concurrent::ConditionImpl, 1228  
   decaf::internal::util::concurrent::MutexImpl, 2742  
   decaf::util::logging::LogWriter, 2376  
 destroyDestination  
   activemq::core::ActiveMQConnection, 251  
 destroyMarshallers  
   activemq::wireformat::openwire::OpenWireFormat, 2840  
 destroyResources  
   decaf::internal::util::ResourceLifecycleManager, 3224  
 DICT  
   inflate.h, 4424  
 DICTID  
   inflate.h, 4424  
 digit  
   decaf::lang::Character, 1072  
 direct  
   gz\_state, 1940  
 DISCONNECT  
   activemq::wireformat::stomp::StompConnector, 3573  
 DiscoveryEvent  
   activemq::commands::DiscoveryEvent, 1723  
   activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1742  
   activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1730  
   activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1734  
   activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1738  
   activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1746  
   activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 1726  
   dispatch  
   activemq::core::ActiveMQConsumer, 287  
   activemq::core::ActiveMQSession, 496  
   activemq::core::Dispatcher, 1750  
   dispatchAsync  
   activemq::commands::ConsumerInfo, 1433  
   activemq::commands::ProducerInfo, 3047  
   DispatchData  
   activemq::core::DispatchData, 1749  
   DIST  
   inflate.h, 4425  
   distbits  
   inflate\_state, 1983  
   distcode  
   inflate\_state, 1984  
   DISTEXT  
   inflate.h, 4425  
   DISTF8  
   infrees.h, 4426  
   dl  
   gz\_data\_s, 1493  
   dmax  
   inflate\_state, 1984  
   doAppendChar  
   decaf::io::Writer, 3954  
   doAppendCharSequence  
   decaf::io::Writer, 3954  
   doAppendCharSequenceStartEnd  
   decaf::io::Writer, 3954  
   doClose  
   activemq::core::ActiveMQConsumer, 287  
   doCreateConnector  
   activemq::transport::failover::FailoverTransportFactory, 1848

activemq::transport::mock::MockTransportFactory, 2735  
 activemq::transport::tcp::SslTransportFactory, 3520  
 activemq::transport::tcp::TcpTransportFactory, 3701  
 doInCms  
   activemq::cmsutil::CmsTemplate::ProducerCallback, 3014  
   activemq::cmsutil::CmsTemplate::ReceiverCallback, 3120  
   activemq::cmsutil::CmsTemplate::SenderCallback, 3291  
   activemq::cmsutil::ProducerCallback, 3014  
   activemq::cmsutil::SessionCallback, 3320  
 DONE  
   inflate.h, 4425  
 done  
   gz\_header\_s, 1939  
 doReadArray  
   decaf::io::FilterInputStream, 1857  
   decaf::io::InputStream, 2005  
   decaf::io::Reader, 3110  
 doReadArrayBounded  
   activemq::io::LoggingInputStream, 2358  
   decaf::internal::net::ssl::openssl::OpenSSLInputStream, 2834  
   decaf::internal::net::tcp::TcpSocketInputStream, 3693  
   decaf::io::BlockingByteArrayInputStream, 802  
   decaf::io::BufferedInputStream, 897  
   decaf::io::ByteArrayInputStream, 989  
   decaf::io::FilterInputStream, 1857  
   decaf::io::InputStream, 2005  
   decaf::io::InputStreamReader, 2015  
   decaf::io::PushbackInputStream, 3090  
   decaf::io::Reader, 3110  
   decaf::util::zip::CheckedInputStream, 1111  
   decaf::util::zip::InflaterInputStream, 1999  
 doReadByte  
   activemq::io::LoggingInputStream, 2358  
   decaf::internal::io::StandardInputStream, 3525  
   decaf::internal::net::ssl::openssl::OpenSSLInputStream, 2834  
   decaf::internal::net::tcp::TcpSocketInputStream, 3693  
   decaf::io::BlockingByteArrayInputStream, 803  
   decaf::io::BufferedInputStream, 897  
   decaf::io::ByteArrayInputStream, 989  
   decaf::io::FilterInputStream, 1857  
   decaf::io::InputStream, 2005  
   decaf::io::PushbackInputStream, 3090  
   decaf::util::zip::CheckedInputStream, 1111  
   decaf::util::zip::InflaterInputStream, 1999  
 doReadDouble  
   decaf::io::Reader, 3110  
 doReadDoubleBounded  
   decaf::nio::DoubleBuffer, 1776  
 doubleToLongBits  
   decaf::lang::Double, 1754  
 doubleToRawLongBits  
   decaf::lang::Double, 1755  
 doubleValue  
   activemq::util::PrimitiveValueNode::PrimitiveValue, 2958  
   decaf::lang::Byte, 922  
   decaf::lang::Character, 1072  
   decaf::lang::Double, 1755  
   decaf::lang::Float, 1868  
   decaf::lang::Integer, 2043  
   decaf::lang::Long, 2382  
   decaf::lang::Number, 2787  
   decaf::lang::Short, 3384  
   decaf::util::concurrent::atomic::AtomicInteger, 710  
 doUnmarshal  
   activemq::wireformat::openwire::OpenWireFormat, 2840  
 doWriteArrayBounded  
   decaf::io::BufferedOutputStream, 900  
   decaf::io::FilterOutputStream, 1863  
   decaf::io::OutputStream, 2858  
   decaf::io::Writer, 3954

activemq::io::LoggingOutputStream, 2360  
 decaf::internal::io::StandardErrorOutputStream, 807  
     3523  
 decaf::internal::io::StandardOutputStream, 3663, 3664  
     3526  
         droppable  
 decaf::internal::net::ssl::openssl::OpenSSLSecureSocketOutputStream, 2491  
     2836  
         dummy  
 decaf::internal::net::tcp::TcpSocketOutputStream, 2082  
     3695  
         duplexConnection  
 decaf::io::BufferedOutputStream, 901  
 decaf::io::ByteArrayOutputStream, 993  
 decaf::io::DataOutputStream, 1548  
 decaf::io::FilterOutputStream, 1863  
 decaf::io::OutputStream, 2858  
 decaf::io::OutputStreamWriter, 2866  
 decaf::io::Writer, 3954  
 decaf::util::zip::CheckedOutputStream, 1113  
 decaf::util::zip::DeflaterOutputStream, 1685  
 doWriteByte  
     activemq::io::LoggingOutputStream, 2360  
     decaf::internal::io::StandardErrorOutputStream, 807  
         3523  
     decaf::internal::io::StandardOutputStream, 3663, 3664  
         3526  
     decaf::internal::net::ssl::openssl::OpenSSLSecureSocketOutputStream, 2491  
         2836  
     decaf::internal::net::tcp::TcpSocketOutputStream, 2082  
         3695  
     decaf::io::BufferedOutputStream, 901  
     decaf::io::ByteArrayOutputStream, 993  
     decaf::io::DataOutputStream, 1548  
     decaf::io::FilterOutputStream, 1863  
     decaf::io::OutputStream, 2859  
     decaf::util::zip::CheckedOutputStream, 1114  
     decaf::util::zip::DeflaterOutputStream, 1685  
 doWriteChar  
     decaf::io::Writer, 3954  
 doWriteString  
     decaf::io::Writer, 3955  
 doWriteStringBounded  
     decaf::io::Writer, 3955  
 doWriteVector  
     decaf::io::Writer, 3955  
 drainPermits  
     decaf::util::concurrent::Semaphore, 3286  
 drainTo  
     decaf::util::concurrent::BlockingQueue, 807  
     decaf::util::concurrent::SynchronousQueue, 3663, 3664  
     decaf::util::concurrent::Message, 2491  
     decaf::internal::nio::ByteBuffer, 970  
     decaf::internal::nio::CharArrayBuffer, 1085  
     decaf::internal::nio::DoubleArrayBuffer, 1770  
     decaf::internal::nio::FloatArrayBuffer, 1884  
     decaf::internal::nio::IntArrayBuffer, 2023  
     decaf::internal::nio::LongArrayBuffer, 2399  
     decaf::internal::nio::ShortArrayBuffer, 3397  
     decaf::nio::ByteBuffer, 1005  
     decaf::nio::CharBuffer, 1097  
     decaf::nio::DoubleBuffer, 1778  
     decaf::nio::FloatBuffer, 1892  
     decaf::nio::IntBuffer, 2031  
     decaf::nio::LongBuffer, 2408  
     decaf::nio::ShortBuffer, 3406  
     DUPS\_OK\_ACKNOWLEDGE  
     decaf::Session, 3308  
     dyn\_dtree  
         internal\_state, 2082  
     dyn\_ltree  
         internal\_state, 2082  
     dyn\_tree  
         tree\_desc\_s, 3840  
     DYN\_TREES  
         zutil.h, 4438  
     dynamicCast  
         decaf::lang::Pointer, 2900  
     DynamicDestinationResolver  
         activemq::cmsutil::DynamicDestinationResolver, 1787  
 E  
     decaf::lang::Math, 2472  
 element  
     decaf::util::AbstractQueue, 166  
     decaf::util::Queue, 3096  
 empty  
     decaf::util::StlQueue, 3559

- encode
  - decaf::net::URLEncoder, 3895
- encodeOthers
  - decaf::internal::net::URLEncoderDecoder, 3866
- end
  - decaf::util::zip::Deflater, 1676
  - decaf::util::zip::Inflater, 1987
- ENOUGH
  - inftrees.h, 4426
- ENOUGH\_DISTS
  - inftrees.h, 4426
- ENOUGH\_LENS
  - inftrees.h, 4426
- enqueue
  - activemq::core::MessageDispatchChannel, 2561
- enqueueFirst
  - activemq::core::MessageDispatchChannel, 2562
- enqueueFront
  - decaf::util::StlQueue, 3559
- enqueueUsage
  - activemq::util::MemoryUsage, 2473
  - activemq::util::Usage, 3896
- ensureCreated
  - decaf::net::ServerSocket, 3297
  - decaf::net::Socket, 3453
- entering
  - decaf::util::logging::Logger, 2349
- Entry
  - decaf::util::Map::Entry, 1789
- eof
  - gz\_state, 1940
- EOFException
  - decaf::io::EOFException, 1790, 1791
- equals
  - activemq::commands::ActiveMQBlobMessage, 174
  - activemq::commands::ActiveMQBytesMessage, 205
  - activemq::commands::ActiveMQDestination, 297
  - activemq::commands::ActiveMQMapMessage, 334
  - activemq::commands::ActiveMQMessage, 370
  - activemq::commands::ActiveMQMessageTemplate, 399
  - activemq::commands::ActiveMQObjectMessage, 415
  - activemq::commands::ActiveMQQueue, 455
  - activemq::commands::ActiveMQStreamMessage, 510
  - activemq::commands::ActiveMQTempDestination, 549
  - activemq::commands::ActiveMQTempQueue, 576
  - activemq::commands::ActiveMQTempTopic, 605
  - activemq::commands::ActiveMQTextMessage, 633
  - activemq::commands::ActiveMQTopic, 662
  - activemq::commands::BaseCommand, 725
  - activemq::commands::BaseDataStructure, 795
  - activemq::commands::BooleanExpression, 817
  - activemq::commands::BrokerId, 830, 831
  - activemq::commands::BrokerInfo, 858
  - activemq::commands::ConnectionControl, 1239
  - activemq::commands::ConnectionError, 1268
  - activemq::commands::ConnectionId, 1299
  - activemq::commands::ConnectionInfo, 1327
  - activemq::commands::ConsumerControl, 1371
  - activemq::commands::ConsumerId, 1400
  - activemq::commands::ConsumerInfo, 1429
  - activemq::commands::ControlCommand, 1461
  - activemq::commands::DataArrayResponse, 1495
  - activemq::commands::DataResponse, 1551
  - activemq::commands::DataStructure, 1630
  - activemq::commands::DestinationInfo, 1693
  - activemq::commands::DiscoveryEvent, 1723
  - activemq::commands::ExceptionResponse, 1723

1803  
 activemq::commands::FlushCommand, 1902  
 activemq::commands::IntegerResponse, 2055  
 activemq::commands::JournalQueueAck, 2117  
 activemq::commands::JournalTopicAck, 2145  
 activemq::commands::JournalTrace, 2173  
 activemq::commands::JournalTransaction, 2200  
 activemq::commands::KeepAliveInfo, 2227  
 activemq::commands::LastPartialCommand, 2261  
 activemq::commands::LocalTransactionId, 2308, 2309  
 activemq::commands::Message, 2481  
 activemq::commands::MessageAck, 2523  
 activemq::commands::MessageDispatch, 2557  
 activemq::commands::MessageDispatchNotification, 2592  
 activemq::commands::MessageId, 2626  
 activemq::commands::MessagePull, 2697  
 activemq::commands::NetworkBridgeFilter, 2747  
 activemq::commands::PartialCommand, 2868  
 activemq::commands::ProducerAck, 2986  
 activemq::commands::ProducerId, 3017  
 activemq::commands::ProducerInfo, 3045  
 activemq::commands::RemoveInfo, 3139  
 activemq::commands::RemoveSubscriptionInfo, 3167  
 activemq::commands::ReplayCommand, 3195  
 activemq::commands::Response, 3229  
 activemq::commands::SessionId, 3322, 3323  
 activemq::commands::SessionInfo, 3350  
 activemq::commands::ShutdownInfo, 3445  
 activemq::commands::SubscriptionInfo, 3618  
 activemq::commands::TransactionId, 3761  
 activemq::commands::TransactionInfo, 3787  
 activemq::commands::WireFormatInfo, 3915  
 activemq::commands::XATransactionId, 3962  
 decaf::lang::Boolean, 813  
 decaf::lang::Byte, 922  
 decaf::lang::Character, 1072  
 decaf::lang::Comparable, 1188  
 decaf::lang::Double, 1755, 1756  
 decaf::lang::Float, 1869  
 decaf::lang::Integer, 2043  
 decaf::lang::Long, 2382  
 decaf::lang::Short, 3384  
 decaf::net::URI, 3858  
 decaf::nio::ByteBuffer, 1005  
 decaf::nio::CharBuffer, 1097  
 decaf::nio::DoubleBuffer, 1779  
 decaf::nio::FloatBuffer, 1892  
 decaf::nio::IntBuffer, 2031  
 decaf::nio::LongBuffer, 2408  
 decaf::nio::ShortBuffer, 3406  
 decaf::security::cert::Certificate, 1056  
 decaf::security::Principal, 2975  
 decaf::util::AbstractCollection, 153  
 decaf::util::Collection, 1160  
 decaf::util::concurrent::ConcurrentStlMap, 1209  
 decaf::util::concurrent::SynchronousQueue, 3665  
 decaf::util::concurrent::TimeUnit, 3751  
 decaf::util::Date, 1635  
 decaf::util::logging::Level, 2292  
 decaf::util::Map, 2423  
 decaf::util::Properties, 3075  
 decaf::util::StlList, 3538  
 decaf::util::StlMap, 3549  
 decaf::util::StlSet, 3569  
 decaf::util::UUID, 3903  
 decaf::io::FileDescriptor, 1852  
 gz\_state, 1940  
 ERR\_MSG  
 zutil.h, 4438  
 ERR\_RETURN  
 zutil.h, 4439  
 Error  
 decaf::util::logging, 144  
 decaf::util::logging::ErrorManager, 1793  
 decaf::util::logging::SimpleLogger, 3445  
 ERROR\_CMD

activemq::wireformat::stomp::StompCommand, 412  
 3573  
 expiration  
 ErrorManager  
 decaf::util::logging::ErrorManager, 1793  
 EXTRA  
 Exception  
 decaf::lang::Exception, 1795, 1796  
 extra  
 exception  
 activemq::commands::ConnectionError, 1269  
 activemq::commands::ExceptionResponse, 1804  
 ExceptionResponse  
 activemq::commands::ExceptionResponse, 1802  
 EXTRA\_STATE  
 deflate.h, 4420  
 ExceptionResponseMarshaller  
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1826  
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1810  
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1814  
 failIfReadOnlyProperties  
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1822  
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1818  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 1805  
 FailoverTransport  
 exclusive  
 activemq::commands::ActiveMQDestination, 303  
 activemq::commands::ConsumerInfo, 1433  
 execute  
 activemq::cmsutil::CmsTemplate, 1144, 1145  
 FAR  
 activemq::core::ActiveMQSessionExecutor, 504  
 fatal  
 decaf::util::concurrent::Executor, 1833  
 executeFirst  
 activemq::core::ActiveMQSessionExecutor, 505  
 fatal  
 decaf::util::logging, 144  
 ExecutionException  
 decaf::util::concurrent::ExecutionException, 1829, 1830  
 exit  
 activemq::commands::ConnectionControl, 1241  
 exiting  
 decaf::util::logging::Logger, 2350  
 EXLEN  
 fd  
 activemq::commands::Message, 2491  
 inflate.h, 4424  
 gz\_header\_s, 1939  
 inflate\_state, 1984  
 extra\_len  
 gz\_header\_s, 1939  
 extra\_max  
 gz\_header\_s, 1939  
 failIfReadOnlyProperties  
 activemq::commands::ActiveMQMessageTemplate, 399  
 activemq::commands::ActiveMQMessageTemplate, 399  
 activemq::commands::ActiveMQMessageTemplate, 399  
 activemq::commands::ActiveMQMessageTemplate, 399  
 activemq::transport::failover::FailoverTransport, 1837  
 FailoverTransportListener  
 activemq::transport::failover::FailoverTransport, 1846  
 activemq::transport::failover::FailoverTransportListener, 1849  
 fatal  
 decaf::util::logging::SimpleLogger, 3445  
 faultTolerant  
 activemq::commands::ConnectionControl, 1241  
 activemq::commands::ConnectionInfo, 1330  
 faultTolerantConfiguration  
 activemq::commands::BrokerInfo, 862  
 fc  
 ct\_data\_s, 1493  
 fd

decaf::net::SocketImpl, 3481	activemq::transport::mock::MockTransport,
gz_state, 1940	2726
FileDescriptor	fireException
decaf::io::FileDescriptor, 1852	activemq::transport::mock::MockTransport,
FileName	2727
activemq::commands::BrokerError::StackFrameElement,	
3521	activemq::commands::MessageAck, 2525
fill	firstNakNumber
decaf::util::zip::InflaterInputStream, 1999	activemq::commands::ReplayCommand,
FILTERED	3197
decaf::util::zip::Deflater, 1681	FLAGS
FilterInputStream	inflate.h, 4424
decaf::io::FilterInputStream, 1856	flags
FilterOutputStream	inflate_state, 1984
decaf::io::FilterOutputStream, 1862	flip
find	decaf::nio::Buffer, 890
decaf::internal::util::TimerTaskHeap, 3746	Float
findFactory	decaf::lang::Float, 1867
activemq::transport::TransportRegistry, 3838	Float_ARRAY
activemq::wireformat::WireFormatRegistry, 3948	activemq::util::PrimitiveValueNode, 2964
FINE	FloatArrayBuffer
decaf::util::logging::Level, 2294	decaf::internal::nio::FloatArrayBuffer, 1880,
fine	1881
decaf::util::logging::Logger, 2350	FloatBuffer
FINER	decaf::nio::FloatBuffer, 1889
decaf::util::logging::Level, 2294	floatToIntBits
finer	decaf::lang::Float, 1869
decaf::util::logging::Logger, 2350	floatToRawIntBits
FINEST	decaf::lang::Float, 1869
decaf::util::logging::Level, 2294	floatValue
finest	activemq::util::PrimitiveValueNode::PrimitiveValue,
decaf::util::logging::Logger, 2351	2958
finish	decaf::lang::Byte, 922
decaf::util::zip::Deflater, 1676	decaf::lang::Character, 1073
decaf::util::zip::DeflaterOutputStream,	decaf::lang::Double, 1756
1685	decaf::lang::Float, 1870
decaf::util::zip::Inflater, 1987	decaf::lang::Integer, 2044
FINISH_STATE	decaf::lang::Long, 2382
deflate.h, 4420	decaf::lang::Number, 2787
finished	decaf::lang::Short, 3384
decaf::util::zip::Deflater, 1676	decaf::util::concurrent::atomic::AtomicInteger,
decaf::util::zip::Inflater, 1988	710
fire	floor
activemq::core::ActiveMQConnection, 252	decaf::lang::Math, 2460
activemq::core::ActiveMQSession, 496	flush
activemq::transport::TransportFilter, 3830	activemq::commands::ConsumerControl,
fireCommand	1373
	decaf::internal::io::StandardErrorOutputStream,
	3523



- decaf::internal::io::StandardOutputStream, 3526
- decaf::io::BufferedOutputStream, 901
- decaf::io::FilterOutputStream, 1863
- decaf::io::Flushable, 1900
- decaf::io::OutputStream, 2859
- decaf::io::OutputStreamWriter, 2866
- decaf::util::logging::Handler, 1942
- decaf::util::logging::StreamHandler, 3599
- FLUSH\_FAILURE
  - decaf::util::logging::ErrorManager, 1793
- FlushCommand
  - activemq::commands::FlushCommand, 1901
- FlushCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1920
  - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1908
  - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1912
  - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1916
  - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1924
  - activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 1904
- format
  - decaf::util::logging::Formatter, 1928
  - decaf::util::logging::SimpleFormatter, 3443
  - decaf::util::logging::XMLFormatter, 3989
- FORMAT\_FAILURE
  - decaf::util::logging::ErrorManager, 1793
- formatId
  - activemq::commands::XATransactionId, 3964
- formatMessage
  - decaf::util::logging::Formatter, 1928
- Freq
  - deflate.h, 4420
- freq
  - ct\_data\_s, 1493
- fromStream
  - activemq::wireformat::stomp::StompFrame, 3578
- fromString
  - decaf::util::UUID, 3903
- front
  - decaf::util::StlQueue, 3559, 3560
- FutureResponse
  - activemq::transport::correlator::FutureResponse, 1933
- GeneralSecurityException
  - decaf::security::GeneralSecurityException, 1935, 1936
- generateId
  - activemq::util::IdGenerator, 1952
- generation
  - decaf::util::concurrent::ConditionHandle, 1227
- GENERIC\_FAILURE
  - decaf::util::logging::ErrorManager, 1793
- GenericResource
  - decaf::internal::util::GenericResource, 1939
- get
  - decaf::internal::nio::DoubleArrayBuffer, 970, 971
  - decaf::internal::nio::FloatArrayBuffer, 1884
  - decaf::internal::nio::LongArrayBuffer, 2400
  - decaf::internal::nio::ShortArrayBuffer, 3397, 3398
- decaf::internal::util::ByteArrayAdapter, 935
- decaf::lang::ArrayPointer, 701
- decaf::lang::Pointer, 2900
- decaf::nio::ByteBuffer, 1005, 1006
- decaf::nio::CharBuffer, 1097, 1098
- decaf::nio::DoubleBuffer, 1779, 1780
- decaf::nio::FloatBuffer, 1892–1894
- decaf::nio::IntBuffer, 2031–2033
- decaf::nio::LongBuffer, 2408–2410
- decaf::nio::ShortBuffer, 3406–3408
- decaf::util::concurrent::atomic::AtomicBoolean, 707
- decaf::util::concurrent::atomic::AtomicInteger, 711
- decaf::util::concurrent::atomic::AtomicReference, 717
- decaf::util::concurrent::ConcurrentStlMap, 1210
- decaf::util::concurrent::Future, 1931
- decaf::util::List, 2298
- decaf::util::Map, 2423, 2424
- decaf::util::StlList, 3538

- decaf::util::StlMap, 3549, 3550
- getAckHandler
  - activemq::commands::Message, 2481
- getAckMode
  - activemq::commands::SessionInfo, 3350
- getAcknowledgeMode
  - activemq::cmsutil::PooledSession, 2916
- getBackup
  - activemq::core::ActiveMQSession, 496
  - cms::Session, 3317
- getAckType
  - activemq::commands::MessageAck, 2523
- getAdditionalPredicate
  - activemq::commands::ConsumerInfo, 1429
- getAddress
  - decaf::net::InetAddress, 1977
- getAdler
  - decaf::util::zip::Deflater, 1676
  - decaf::util::zip::Inflater, 1988
- getAlgorithm
  - decaf::security::Key, 2254
- getAndAdd
  - decaf::util::concurrent::atomic::AtomicInteger, 711
- getAndDecrement
  - decaf::util::concurrent::atomic::AtomicInteger, 711
- getAndIncrement
  - decaf::util::concurrent::atomic::AtomicInteger, 711
- getAndSet
  - decaf::util::concurrent::atomic::AtomicBoolean, 707
  - decaf::util::concurrent::atomic::AtomicInteger, 711
  - decaf::util::concurrent::atomic::AtomicReference, 717
- getAnonymousLogger
  - decaf::util::logging::Logger, 2351
- getAnyAddress
  - decaf::net::InetAddress, 1977
- getAprPool
  - decaf::internal::AprPool, 697
- getArrival
  - activemq::commands::Message, 2482
- getAuthority
  - decaf::internal::net::URIType, 3885
  - decaf::net::URI, 3858
- getBacklog
  - decaf::util::concurrent::ThreadPool, 3720
- getBackOffMultiplier
  - activemq::core::policies::DefaultRedeliveryPolicy, 1645
  - activemq::core::RedeliveryPolicy, 3123
  - activemq::transport::failover::FailoverTransport, 1838
- getBackup
  - activemq::transport::failover::BackupTransportPool, 721
- getBackupPoolSize
  - activemq::transport::failover::BackupTransportPool, 722
- getBasicConstraints
  - decaf::security::cert::X509Certificate, 3959
- getBlockSize
  - decaf::util::concurrent::ThreadPool, 3720
- getBody
  - activemq::wireformat::stomp::StompFrame, 3578, 3579
- getBodyBytes
  - activemq::commands::ActiveMQBytesMessage, 206
  - cms::BytesMessage, 1026
- getBodyLength
  - activemq::commands::ActiveMQBytesMessage, 206
  - activemq::wireformat::stomp::StompFrame, 3579
  - cms::BytesMessage, 1027
- getBool
  - activemq::util::PrimitiveList, 2931
  - activemq::util::PrimitiveMap, 2943
  - activemq::util::PrimitiveValueNode, 2967
- getBoolean
  - activemq::commands::ActiveMQMapMessage, 334
  - cms::MapMessage, 2434
- getBooleanProperty
  - activemq::commands::ActiveMQMessageTemplate, 399
- getBrokerId
  - activemq::wireformat::openwire::utils::MessagePropertyInterface, 2691
  - cms::Message, 2498
- getBranchQualifier
  - activemq::commands::XATransactionId, 3963

activemq::commands::BrokerInfo, 858, getBytes  
 859  
 getBrokerInTime  
 activemq::commands::Message, 2482  
 getBrokerName  
 activemq::commands::BrokerInfo, 859  
 activemq::commands::DiscoveryEvent,  
 1724  
 getBrokerOutTime  
 activemq::commands::Message, 2482  
 getBrokerPath  
 activemq::commands::ConnectionInfo,  
 1327  
 activemq::commands::ConsumerInfo,  
 1429  
 activemq::commands::DestinationInfo,  
 1694  
 activemq::commands::Message, 2482  
 activemq::commands::ProducerInfo, 3045  
 getBrokerSequencedId  
 activemq::commands::MessageId, 2626  
 getBrokerUploadUrl  
 activemq::commands::BrokerInfo, 859  
 getBrokerURL  
 activemq::commands::BrokerInfo, 859  
 activemq::core::ActiveMQConnection,  
 252  
 activemq::core::ActiveMQConnectionFactory,  
 269  
 getByAddress  
 decaf::net::InetAddress, 1977, 1978  
 getByte  
 activemq::commands::ActiveMQMapMessage,  
 335  
 activemq::util::PrimitiveList, 2932  
 activemq::util::PrimitiveMap, 2944  
 activemq::util::PrimitiveValueNode, 2967  
 cms::MapMessage, 2434  
 getByteArray  
 activemq::util::PrimitiveList, 2932  
 activemq::util::PrimitiveMap, 2944  
 activemq::util::PrimitiveValueNode, 2967  
 decaf::internal::util::ByteArrayAdapter,  
 936  
 getByteProperty  
 activemq::commands::ActiveMQMessageTemplate,  
 400  
 activemq::wireformat::openwire::utils::MessageID,  
 2691  
 cms::Message, 2499  
 activemq::commands::ActiveMQMapMessage,  
 335  
 cms::MapMessage, 2434  
 getBytesRead  
 decaf::util::zip::Deflater, 1677  
 decaf::util::zip::Inflater, 1988  
 getBytesWritten  
 decaf::util::zip::Deflater, 1677  
 decaf::util::zip::Inflater, 1988  
 getCacheSize  
 activemq::commands::WireFormatInfo,  
 3916  
 activemq::wireformat::openwire::OpenWireFormat,  
 2841  
 getCapacity  
 decaf::internal::util::ByteArrayAdapter,  
 936  
 getCause  
 activemq::commands::BrokerError, 825  
 cms::CMSEException, 1132  
 decaf::lang::Exception, 1797  
 decaf::lang::Throwable, 3726  
 getChar  
 activemq::commands::ActiveMQMapMessage,  
 336  
 activemq::util::PrimitiveList, 2933  
 activemq::util::PrimitiveMap, 2945  
 activemq::util::PrimitiveValueNode, 2968  
 cms::MapMessage, 2435  
 decaf::internal::nio::ByteBuffer, 971,  
 972  
 decaf::internal::util::ByteArrayAdapter,  
 936  
 decaf::nio::ByteBuffer, 1007  
 getCharArray  
 decaf::internal::util::ByteArrayAdapter,  
 937  
 getCharCapacity  
 decaf::internal::util::ByteArrayAdapter,  
 937  
 getChecksum  
 decaf::util::zip::CheckedInputStream, 1111  
 decaf::util::zip::CheckedOutputStream,  
 1114  
 getCertificates  
 decaf::net::ssl::SSLParameters, 3496  
 getClientPropertyInterceptor,  
 activemq::core::ActiveMQConnection,  
 252

cms::Connection, 1235  
 getClientId  
 activemq::commands::ActiveMQDestination, 297  
 activemq::commands::ConnectionInfo, 1327  
 activemq::commands::JournalTopicAck, 2145  
 activemq::commands::RemoveSubscriptionInfo, 3167  
 activemq::commands::SubscriptionInfo, 3618  
 activemq::core::ActiveMQConnectionFactory, 269  
 getCloseTimeout  
 activemq::core::ActiveMQConnection, 253  
 activemq::core::ActiveMQConnectionFactory, 269  
 getCluster  
 activemq::commands::Message, 2482  
 getCMSCorrelationID  
 activemq::commands::ActiveMQMessageTemplate, 400  
 cms::Message, 2500  
 getCMSDeliveryMode  
 activemq::commands::ActiveMQMessageTemplate, 400  
 cms::Message, 2500  
 getCMSDestination  
 activemq::commands::ActiveMQDestination, 298  
 activemq::commands::ActiveMQMessageTemplate, 401  
 activemq::commands::ActiveMQQueue, 455  
 activemq::commands::ActiveMQTempQueue, 576  
 activemq::commands::ActiveMQTempTopic, 605  
 activemq::commands::ActiveMQTopic, 662  
 cms::Message, 2501  
 getCMSExpiration  
 activemq::commands::ActiveMQMessageTemplate, 401  
 cms::Message, 2501  
 getCMSMajorVersion  
 activemq::core::ActiveMQConnectionMetaData, 276  
 cms::Connection, 1235  
 getCMSMessageID  
 activemq::commands::ActiveMQMessageTemplate, 401  
 cms::Message, 2502  
 getCMSMinorVersion  
 activemq::core::ActiveMQConnectionMetaData, 276  
 cms::ConnectionMetaData, 1356  
 getCMSPriority  
 activemq::commands::ActiveMQMessageTemplate, 402  
 cms::Message, 2503  
 getCMSProperties  
 activemq::commands::ActiveMQQueue, 455  
 activemq::commands::ActiveMQTempQueue, 577  
 activemq::commands::ActiveMQTempTopic, 605  
 activemq::commands::ActiveMQTopic, 662  
 cms::Message, 2500  
 getCMSProviderName  
 activemq::core::ActiveMQConnectionMetaData, 276  
 cms::ConnectionMetaData, 1356  
 getCMSRedelivered  
 activemq::commands::ActiveMQMessageTemplate, 402  
 cms::Message, 2503  
 getCMSReplyTo  
 activemq::commands::ActiveMQMessageTemplate, 402  
 cms::Message, 2504  
 getCMSTimestamp  
 activemq::commands::ActiveMQMessageTemplate, 403  
 cms::Message, 2504  
 getCMSType  
 activemq::commands::ActiveMQMessageTemplate, 403  
 cms::Message, 2505  
 getCMSVersion  
 activemq::core::ActiveMQConnectionMetaData, 277  
 cms::ConnectionMetaData, 1357  
 getCMSXPropertyNames  
 activemq::core::ActiveMQConnectionMetaData, 277

---

cms::ConnectionMetaData, 1357  
 getCollisionAvoidancePercent  
     activemq::core::policies::DefaultRedeliveryPolicy, 1371  
     1645  
     activemq::core::RedeliveryPolicy, 3124  
 getCommand  
     activemq::commands::ControlCommand, 1461  
     activemq::wireformat::stomp::StompFrame, 3579  
 getCommandId  
     activemq::commands::BaseCommand, 726  
     activemq::commands::Command, 1166  
     activemq::commands::PartialCommand, 2868  
 getCommands  
     activemq::state::TransactionState, 3814  
 getComponents  
     activemq::util::CompositeData, 1192  
 getConnectedBrokers  
     activemq::commands::ConnectionControl, 1239  
 getConnection  
     activemq::commands::Message, 2482  
     activemq::core::ActiveMQSession, 497  
 getConnectionFactory  
     activemq::cmsutil::CmsAccessor, 1126  
 getConnectionId  
     activemq::commands::BrokerInfo, 859  
     activemq::commands::ConnectionError, 1268  
     activemq::commands::ConnectionInfo, 1327  
     activemq::commands::ConsumerId, 1400  
     activemq::commands::DestinationInfo, 1694  
     activemq::commands::LocalTransactionId, 2309  
     activemq::commands::ProducerId, 3017  
     activemq::commands::RemoveSubscriptionInfo, 3167  
     activemq::commands::SessionId, 3323  
     activemq::commands::TransactionInfo, 3787  
     activemq::core::ActiveMQConnection, 253  
 getConnectionInfo  
     activemq::core::ActiveMQConnection, 253  
     getConsumerId  
         activemq::commands::ConsumerControl, 1371  
         activemq::commands::ConsumerInfo, 1429  
         activemq::commands::MessageAck, 2523  
         activemq::commands::MessageDispatch, 2557  
         activemq::commands::MessageDispatchNotification, 2592, 2593  
         activemq::commands::MessagePull, 2697  
         activemq::core::ActiveMQConsumer, 287  
         activemq::core::DispatchData, 1749  
     getConsumerInfo  
         activemq::core::ActiveMQConsumer, 287  
     getConsumerState  
         activemq::state::SessionState, 3379  
     getConsumerStates  
         activemq::state::SessionState, 3379  
     getContent  
         activemq::commands::Message, 2482  
     getContext  
         decaf::internal::net::ssl::DefaultSSLContext, 1658  
     getCorrelationId  
         activemq::commands::Message, 2482, 2483  
         activemq::commands::MessagePull, 2697  
         activemq::commands::Response, 3229  
     getCount  
         decaf::util::concurrent::CountDownLatch, 1489  
     getData  
         activemq::commands::DataArrayResponse, 1495  
         activemq::commands::DataResponse, 1552  
         activemq::commands::PartialCommand, 2868, 2869  
     getDataStructure  
         activemq::commands::Message, 2483  
     getDataStructureType  
         activemq::commands::ActiveMQBlobMessage, 175  
         activemq::commands::ActiveMQBytesMessage, 207  
         activemq::commands::ActiveMQDestination, 298  
         activemq::commands::ActiveMQMapMessage, 336

activemq::commands::ActiveMQMessage,	2056
370	activemq::commands::JournalQueueAck,
activemq::commands::ActiveMQObjectMessage,	2118
416	activemq::commands::JournalTopicAck,
activemq::commands::ActiveMQQueue,	2145
456	activemq::commands::JournalTrace,
activemq::commands::ActiveMQStreamMessage,	2173
510	activemq::commands::JournalTransaction,
activemq::commands::ActiveMQTempDestination,	2200
549	activemq::commands::KeepAliveInfo,
activemq::commands::ActiveMQTempQueue,	2227
577	activemq::commands::LastPartialCommand,
activemq::commands::ActiveMQTempTopic,	2262
605	activemq::commands::LocalTransactionId,
activemq::commands::ActiveMQTextMessage,	2309
633	activemq::commands::Message,
activemq::commands::ActiveMQTopic,	2483
662	activemq::commands::MessageAck,
activemq::commands::BrokerError,	2523
825	activemq::commands::MessageDispatch,
activemq::commands::BrokerId,	2557
831	activemq::commands::MessageDispatchNotification,
activemq::commands::BrokerInfo,	2593
859	activemq::commands::MessageId,
activemq::commands::ConnectionControl,	2626
1239	activemq::commands::MessagePull,
activemq::commands::ConnectionError,	2698
1268	activemq::commands::NetworkBridgeFilter,
activemq::commands::ConnectionId,	2748
1299	activemq::commands::PartialCommand,
activemq::commands::ConnectionInfo,	2869
1327	activemq::commands::ProducerAck,
activemq::commands::ConsumerControl,	2986
1371	activemq::commands::ProducerId,
activemq::commands::ConsumerId,	3017
1400	activemq::commands::ProducerInfo,
activemq::commands::ConsumerInfo,	3045
1430	activemq::commands::RemoveInfo,
activemq::commands::ControlCommand,	3139
1461	activemq::commands::RemoveSubscriptionInfo,
activemq::commands::DataArrayResponse,	3167
1495	activemq::commands::ReplayCommand,
activemq::commands::DataResponse,	3195
1552	activemq::commands::Response,
activemq::commands::DataStructure,	3229
1631	activemq::commands::SessionId,
activemq::commands::DestinationInfo,	3323
1694	activemq::commands::SessionInfo,
activemq::commands::DiscoveryEvent,	3350
1724	activemq::commands::ShutdownInfo,
activemq::commands::ExceptionResponse,	3414
1803	activemq::commands::SubscriptionInfo,
activemq::commands::FlushCommand,	3618
1902	activemq::commands::TransactionId,
activemq::commands::IntegerResponse,	3762
	activemq::commands::TransactionInfo,
	3787
	activemq::commands::WireFormatInfo,
	3916
	activemq::commands::XATransactionId,
	3963
	activemq::wireformat::openwire::marshal::DataStreamMarshal
	1585
	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMe
	183

activemq::wireformat::openwire::marshal::v1::ActiveMQByteMessageMarshaller,	226
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller,	350
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,	376
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller,	422
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,	465
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller,	528
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller,	584
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller,	616
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller,	645
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,	673
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessageDispatchMarshaller,	841
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessageDispatchNotificationMarshaller,	872
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessageIdMarshaller,	1251
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessagePullMarshaller,	1283
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::NetworkBridgeFilterMarshaller,	1314
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::PartialCommandMarshaller,	1344
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ProducerAckMarshaller,	1387
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ProducerIdMarshaller,	1415
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ProducerInfoMarshaller,	1448
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::RemoveInfoMarshaller,	1476
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller,	1509
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ReplayCommandMarshaller,	1574
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ResponseMarshaller,	1709
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::SessionIdMarshaller,	1742
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::SessionInfoMarshaller,	1826
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::FlushCommandMarshaller,	1920
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::IntegerResponseMarshaller,	2074
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::JournalQueueAckMarshaller,	2140
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::JournalTopicAckMarshaller,	2169
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::JournalTraceMarshaller,	2191
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::JournalTransactionMarshaller,	2223
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::KeepAliveInfoMarshaller,	2250
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::LastPartialCommandMarshaller,	2284
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::LocalTransactionIdMarshaller,	2332
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessageAckMarshaller,	2543
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessageDispatchMarshaller,	2584
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessageDispatchNotificationMarshaller,	2613
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessageIdMarshaller,	2649
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::MessagePullMarshaller,	2717
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::NetworkBridgeFilterMarshaller,	2770
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::PartialCommandMarshaller,	2893
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ProducerAckMarshaller,	3009
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ProducerIdMarshaller,	3040
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ProducerInfoMarshaller,	3057
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::RemoveInfoMarshaller,	3154
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller,	3171
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ReplayCommandMarshaller,	3202
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::ResponseMarshaller,	3256
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::SessionIdMarshaller,	3345
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormat::openwire::marshal::v1::SessionInfoMarshaller,	3361

activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatOpenwire::marshal::v2::ControlCommandMarshaller 1464  
 3425  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatOpenwire::marshal::v2::DataArrayResponseMarshaller 1497  
 3625  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatOpenwire::marshal::v2::DataResponseMarshaller 1562  
 3794  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatOpenwire::marshal::v2::DestinationInfoMarshaller 1697  
 3940  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatOpenwire::marshal::v2::DiscoveryEventManager 1730  
 3978  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBinaryMessageMarshaller 1810  
 191  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBinaryMessageMarshaller 1908  
 242  
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller 2062  
 362  
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller 2124  
 388  
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller 2153  
 434  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller 2176  
 477  
 activemq::wireformat::openwire::marshal::v2::ActiveMQSimpleMessageMarshaller 2207  
 540  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller 2234  
 596  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller 2272  
 624  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller 2316  
 657  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller 2531  
 685  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::MessageDispatcher 2568  
 853  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::MessageDispatcher 2600  
 884  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::MessageIdMarshaller 2629  
 1263  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::MessagePullMarshaller 2701  
 1271  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::NetworkBridgeFilter 2750  
 1302  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::PartialCommandMarshaller 2876  
 1332  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::ProducerAckMarshaller 2989  
 1375  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::ProducerIdMarshaller 3020  
 1403  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenwire::marshal::v2::ProducerInfoMarshaller 3053  
 1436



activemq::wireformat::openwire::marshal::v2::ActiveMQInfoMarshaller,	3142	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller,	1275
activemq::wireformat::openwire::marshal::v2::ActiveMQSubscriptionInfoMarshaller,	3179	activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller,	1306
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3206	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller,	1336
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3242	activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller,	1379
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3326	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,	1407
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3369	activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,	1440
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3421	activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller,	1468
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3641	activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller,	1501
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3810	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller,	1566
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3932	activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller,	1701
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3970	activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller,	1734
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	179	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller,	1814
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	222	activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller,	1912
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	346	activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller,	2066
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	372	activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller,	2132
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	418	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller,	2157
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	461	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,	2179
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	524	activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller,	2211
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	580	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller,	2238
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	608	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller,	2268
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	637	activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller,	2320
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	665	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller,	2535
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	833	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller,	2572
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	864	activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller,	2605
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	1243	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	2641

activemq::wireformat::openwire::marshal::v3::ActiveMQTempTo  
 2709 612  
 activemq::wireformat::openwire::marshal::v3::ActiveMQBridgeFormatMarshaller; marshal::v4::ActiveMQTextMe  
 2762 641  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ActiveMQTopicM  
 2884 669  
 activemq::wireformat::openwire::marshal::v3::ActiveMQAddFormatMarshaller; marshal::v4::BrokerIdMarshall  
 2997 837  
 activemq::wireformat::openwire::marshal::v3::ActiveMQInfoMarshaller; marshal::v4::BrokerInfoMarsha  
 3028 868  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ConnectionContr  
 3065 1247  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ConnectionErrorM  
 3150 1279  
 activemq::wireformat::openwire::marshal::v3::ActiveMQSubscriptionInfoMarshaller; marshal::v4::ConnectionIdMar  
 3175 1310  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ConnectionInfoM  
 3210 1340  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ConsumerContro  
 3252 1383  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ConsumerIdMars  
 3341 1411  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ConsumerInfoMa  
 3365 1444  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::ControlCommand  
 3433 1472  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::DataArrayRespon  
 3621 1505  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::DataResponseM  
 3798 1570  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::DestinationInfoM  
 3944 1705  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormatMarshaller; marshal::v4::DiscoveryEventM  
 3982 1738  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageMarshaller; marshal::v4::ExceptionRespon  
 187 1822  
 activemq::wireformat::openwire::marshal::v4::ActiveMQBinaryMessageMarshaller; marshal::v4::FlushCommandM  
 230 1916  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller; marshal::v4::IntegerResponse  
 354 2070  
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller; marshal::v4::JournalQueueAck  
 380 2136  
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller; marshal::v4::JournalTopicAckM  
 426 2165  
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller; marshal::v4::JournalTraceMar  
 469 2187  
 activemq::wireformat::openwire::marshal::v4::ActiveMQSimpleMessageMarshaller; marshal::v4::JournalTransactio  
 532 2219  
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller; marshal::v4::KeepAliveInfoMa  
 588 2242

activemq::wireformat::openwire::marshal::v4::ActiveMQFormatMessageMarshaller	2280	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	358
activemq::wireformat::openwire::marshal::v4::ActiveMQTransactionIdMarshaller	2328	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	384
activemq::wireformat::openwire::marshal::v4::ActiveMQAcknowledge	2539	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	430
activemq::wireformat::openwire::marshal::v4::ActiveMQDispatchInfoMarshaller	2580	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	473
activemq::wireformat::openwire::marshal::v4::ActiveMQDispatchNotificationMarshaller	2609	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	536
activemq::wireformat::openwire::marshal::v4::ActiveMQIdMarshaller	2633	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	592
activemq::wireformat::openwire::marshal::v4::ActiveMQRuleMarshaller	2713	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	620
activemq::wireformat::openwire::marshal::v4::ActiveMQBridgeFilterMarshaller	2766	activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	649
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueIdMarshaller	2889	activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	677
activemq::wireformat::openwire::marshal::v4::ActiveMQAcknowledge	2993	activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	845
activemq::wireformat::openwire::marshal::v4::ActiveMQIdMarshaller	3024	activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	876
activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMarshaller	3049	activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	1255
activemq::wireformat::openwire::marshal::v4::ActiveMQWireMarshaller	3162	activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	1287
activemq::wireformat::openwire::marshal::v4::ActiveMQSubscriptionInfoMarshaller	3191	activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	1318
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueIdMarshaller	3198	activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	1348
activemq::wireformat::openwire::marshal::v4::ActiveMQResponseMarshaller	3238	activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	1391
activemq::wireformat::openwire::marshal::v4::ActiveMQIdMarshaller	3329	activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	1419
activemq::wireformat::openwire::marshal::v4::ActiveMQInfoMarshaller	3373	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	1452
activemq::wireformat::openwire::marshal::v4::ActiveMQWireMarshaller	3437	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller	1480
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	3633	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	1513
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	3806	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	1554
activemq::wireformat::openwire::marshal::v4::ActiveMQAcknowledge	3936	activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	1717
activemq::wireformat::openwire::marshal::v4::ActiveMQTransactionIdMarshaller	3974	activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller	1746
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	195	activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	1818
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	234	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	1924

activemq::wireformat::openwire::marshal::v5::ActiveMQResponseMarshaller 3629  
 2078  
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueIdMarshaller 3790  
 2128  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicIdMarshaller 3924  
 2149  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTraceMarshaller 3986  
 2195  
 activemq::wireformat::openwire::marshal::v5::ActiveMQTransactionalMarshaller 199  
 2215  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 238  
 2246  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 366  
 2276  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 392  
 2324  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 438  
 2547  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 481  
 2576  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 544  
 2617  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 600  
 2637  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 628  
 2705  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 653  
 2758  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 681  
 2880  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 849  
 3001  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 880  
 3032  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1259  
 3061  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1291  
 3158  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1322  
 3187  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1352  
 3218  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1395  
 3247  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1423  
 3337  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1456  
 3357  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller 1484  
 3429

activemq::wireformat::openwire::marshal::v6::DeleteResponseMarshaller, 3183  
 1517  
 activemq::wireformat::openwire::marshal::v6::DeleteResponseMarshaller, 3214  
 1558  
 activemq::wireformat::openwire::marshal::v6::DestinationInfoMarshaller, 3261  
 1713  
 activemq::wireformat::openwire::marshal::v6::DiscoveryEventMarshaller, 3333  
 1726  
 activemq::wireformat::openwire::marshal::v6::ExceptionResponseMarshaller, 3353  
 1806  
 activemq::wireformat::openwire::marshal::v6::FlushCommandMarshaller, 3417  
 1904  
 activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller, 3637  
 2058  
 activemq::wireformat::openwire::marshal::v6::MessageQueueAckMarshaller, 3802  
 2121  
 activemq::wireformat::openwire::marshal::v6::MessageTopicAckMarshaller, 3928  
 2161  
 activemq::wireformat::openwire::marshal::v6::MessageTraceMarshaller, 3966  
 2183  
 activemq::wireformat::openwire::marshal::v6::DefaultTransactionMarshaller, 3304  
 2203  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3471  
 2230  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3490  
 2264  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3505  
 2312  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3517  
 2527  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3298  
 2588  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1662  
 2596  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1669  
 2645  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 2807  
 2721  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 2831  
 2754  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3505  
 2871  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3517  
 3005  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1145  
 3036  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1145  
 3069  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3490  
 3146  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1687  
 3183  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1687  
 3214  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3261  
 3261  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3333  
 3333  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3353  
 3353  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3417  
 3417  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3637  
 3637  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3802  
 3802  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3928  
 3928  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3966  
 3966  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3304  
 3304  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3471  
 3471  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3490  
 3490  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3505  
 3505  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3517  
 3517  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3298  
 3298  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1662  
 1662  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1669  
 1669  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 2807  
 2807  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 2831  
 2831  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3505  
 3505  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3517  
 3517  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1145  
 1145  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1145  
 1145  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 3490  
 3490  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1687  
 1687  
 activemq::wireformat::openwire::marshal::v6::DefaultSSLContext, 1687  
 1687

activemq::cmsutil::CachedProducer, 1046  
 activemq::cmsutil::CmsTemplate, 1146  
 activemq::core::ActiveMQProducer, 443  
 cms::MessageProducer, 2683  
 getDeliverySequenceId  
 activemq::commands::MessageDispatchNotification, 2593  
 getDestination  
 activemq::cmsutil::CmsTemplate::ProducerExecutor, 3014  
 activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3120  
 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3222  
 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3223  
 activemq::commands::ConsumerControl, 1371  
 activemq::commands::ConsumerInfo, 1430  
 activemq::commands::DestinationInfo, 1694  
 activemq::commands::JournalQueueAck, 2118  
 activemq::commands::JournalTopicAck, 2146  
 activemq::commands::Message, 2483  
 activemq::commands::MessageAck, 2523  
 activemq::commands::MessageDispatch, 2557  
 activemq::commands::MessageDispatchNotification, 2593  
 activemq::commands::MessagePull, 2698  
 activemq::commands::ProducerInfo, 3045  
 activemq::commands::SubscriptionInfo, 3618  
 getDestinationResolver  
 activemq::cmsutil::CmsDestinationAccessor, 1129  
 getDestinationType  
 activemq::commands::ActiveMQDestination, 298  
 activemq::commands::ActiveMQQueue, 456  
 activemq::commands::ActiveMQTempQueue, 577  
 activemq::commands::ActiveMQTempTopic, 606  
 activemq::commands::ActiveMQTopic, 663  
 cms::Destination, 1690  
 getDisableMessageID  
 activemq::cmsutil::CachedProducer, 1046  
 activemq::core::ActiveMQProducer, 443  
 cms::MessageProducer, 2683  
 getDisableMessageTimeStamp  
 activemq::cmsutil::CachedProducer, 1046  
 activemq::core::ActiveMQProducer, 443  
 cms::MessageProducer, 2683  
 getDouble  
 activemq::commands::ActiveMQMapMessage, 336  
 activemq::util::PrimitiveList, 2933  
 activemq::util::PrimitiveMap, 2945  
 activemq::util::PrimitiveValueNode, 2968  
 cms::MapMessage, 2435  
 decaf::internal::nio::ByteBuffer, 972  
 decaf::internal::util::ByteArrayAdapter, 937  
 decaf::nio::ByteBuffer, 1008  
 getDoubleArray  
 decaf::internal::util::ByteArrayAdapter, 937  
 getDoubleAt  
 decaf::internal::util::ByteArrayAdapter, 938  
 getDoubleCapacity  
 decaf::internal::util::ByteArrayAdapter, 938  
 getDoubleProperty  
 activemq::commands::ActiveMQMessageTemplate, 403  
 activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2691  
 cms::Message, 2506  
 getDurableTopicPrefetch  
 activemq::core::policies::DefaultPrefetchPolicy, 1641  
 activemq::core::PrefetchPolicy, 2926  
 getEnabledCipherSuites  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2800  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 2814  
 decaf::net::ssl::SSLServerSocket, 3501  
 decaf::net::ssl::SSLSocket, 3510  
 getEnabledProtocols

- decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796
- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2800
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2814
- decaf::net::ssl::SSLServerSocket, 3501
- decaf::net::ssl::SSLSocket, 3510
- getEncoded
  - decaf::security::auth::x500::X500Principal, 3957
  - decaf::security::cert::Certificate, 1056
  - decaf::security::Key, 2254
- getEnumeration
  - activemq::core::ActiveMQQueueBrowser, 458
  - cms::QueueBrowser, 3099
- getenv
  - decaf::lang::System, 3674, 3675
- getErrorCode
  - decaf::net::SocketError, 3464
- getErrorManager
  - decaf::util::logging::Handler, 1942
- getErrorString
  - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2824
  - decaf::net::SocketError, 3464
- getException
  - activemq::commands::ConnectionError, 1268
  - activemq::commands::ExceptionResponse, 1804
- getExceptionClass
  - activemq::commands::BrokerError, 825
- getExceptionListener
  - activemq::core::ActiveMQConnection, 253
  - activemq::core::ActiveMQConnectionFactory, 269
  - activemq::core::ActiveMQSession, 497
  - cms::Connection, 1235
- getExpiration
  - activemq::commands::Message, 2483
- getFileDescriptor
  - decaf::net::SocketImpl, 3476
- getFilter
  - decaf::util::logging::Handler, 1943
  - decaf::util::logging::Logger, 2351
- getFirstMessageId
  - activemq::commands::MessageAck, 2523, 2524
  - activemq::commands::ReplayCommand, 3196
  - activemq::commands::ActiveMQMapMessage, 337
  - activemq::util::PrimitiveList, 2934
  - activemq::util::PrimitiveMap, 2946
  - activemq::util::PrimitiveValueNode, 2968
  - cms::MapMessage, 2436
  - decaf::internal::nio::ByteBuffer, 973
  - decaf::internal::util::ByteArrayAdapter, 938
  - decaf::nio::ByteBuffer, 1009
  - decaf::internal::util::ByteArrayAdapter, 939
  - decaf::internal::util::ByteArrayAdapter, 939
  - decaf::internal::util::ByteArrayAdapter, 939
  - decaf::internal::util::ByteArrayAdapter, 939
  - activemq::commands::ActiveMQMessageTemplate, 404
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2692
  - cms::Message, 2506
  - decaf::security::Key, 2254
  - activemq::commands::XATransactionId, 3963
  - decaf::util::logging::Handler, 1943
  - activemq::util::CompositeData, 1192
  - decaf::internal::net::URIType, 3886
  - decaf::net::URI, 3858
  - decaf::util::concurrent::ThreadPool, 3721
  - decaf::internal::AprPool, 697
  - decaf::internal::DecafRuntime, 1638
  - activemq::commands::XATransactionId, 3963

getGroupID  
     activemq::commands::Message, 2483  
 getGroupSequence  
     activemq::commands::Message, 2483  
 getHandlers  
     decaf::util::logging::Logger, 2351  
 getHead  
     decaf::util::logging::Formatter, 1928  
     decaf::util::logging::XMLFormatter, 3989  
 getHoldCount  
     decaf::util::concurrent::locks::ReentrantLock, 3128  
     decaf::util::concurrent::ThreadPool, 3721  
     decaf::util::logging::LogWriter, 2376  
 getHost  
     activemq::util::CompositeData, 1192  
     decaf::internal::net::URIType, 3886  
     decaf::net::URI, 3858  
 getHostAddress  
     decaf::net::InetAddress, 1978  
 getHostName  
     decaf::net::InetAddress, 1978  
 getHostname  
     activemq::util::IdGenerator, 1952  
 getId  
     activemq::state::TransactionState, 3814  
     decaf::lang::Thread, 3711  
 getIndex  
     decaf::net::URISyntaxException, 3883  
 getInetAddress  
     decaf::net::Socket, 3453  
     decaf::net::SocketImpl, 3477  
 getInfo  
     activemq::state::ConnectionState, 1360  
     activemq::state::ConsumerState, 1459  
     activemq::state::ProducerState, 3072  
     activemq::state::SessionState, 3379  
 getInitialDelayTime  
     activemq::transport::inactivity::InactivityMonitor, 1966  
 getInitialReconnectDelay  
     activemq::transport::failover::FailoverTransport, 1838  
 getInitialRedeliveryDelay  
     activemq::core::policies::DefaultRedeliveryPolicy, 1645  
     activemq::core::RedeliveryPolicy, 3124  
 getInput  
     decaf::net::URISyntaxException, 3883  
 getInputStream  
     decaf::internal::net::ssl::openssl::OpenSSLSSLContext, 2814  
     decaf::internal::net::tcp::TcpSocket, 3687  
     decaf::net::Socket, 3453  
     decaf::net::SocketImpl, 3477  
 getInstance  
     activemq::transport::mock::MockTransport, 2727  
     activemq::transport::TransportRegistry, 3839  
     activemq::wireformat::WireFormatRegistry, 3949  
 getInt  
     activemq::commands::ActiveMQMapMessage, 337  
     activemq::util::PrimitiveList, 2934  
     activemq::util::PrimitiveMap, 2946  
     activemq::util::PrimitiveValueNode, 2969  
     cms::MapMessage, 2436  
     decaf::internal::nio::ByteBuffer, 973, 974  
     decaf::internal::util::ByteArrayAdapter, 940  
     decaf::nio::ByteBuffer, 1009, 1010  
 getIntArray  
     decaf::internal::util::ByteArrayAdapter, 940  
 getIntAt  
     decaf::internal::util::ByteArrayAdapter, 940  
 getIntCapacity  
     decaf::internal::util::ByteArrayAdapter, 941  
 getIntProperty  
     activemq::commands::ActiveMQMessageTemplate, 404  
     activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2692  
     cms::Message, 2507  
 getJmsConsumerUniqueID  
     decaf::security::cert::X509Certificate, 3959  
 getJmsPrincipal  
     decaf::security::cert::X509Certificate, 3959  
 getKeepAlive  
     decaf::net::Socket, 3454  
 getKey  
     decaf::util::Map::Entry, 1789  
 getKeyUsage



- decaf::security::cert::X509Certificate, 3959
- getLastDeliveredSequenceId
  - activemq::commands::RemoveInfo, 3140
  - activemq::core::ActiveMQConsumer, 287
  - activemq::core::ActiveMQSession, 497
- getLastMessageId
  - activemq::commands::MessageAck, 2524
- getLastNakNumber
  - activemq::commands::ReplayCommand, 3196
- getLastSequenceId
  - activemq::util::LongSequenceGenerator, 2416
- getLeastSignificantBits
  - decaf::util::UUID, 3903
- getLevel
  - decaf::util::logging::Handler, 1943
  - decaf::util::logging::Logger, 2352
  - decaf::util::logging::LogRecord, 2371
- getLimit
  - activemq::util::MemoryUsage, 2474
- getList
  - activemq::util::PrimitiveValueNode, 2969
- getLocalAddress
  - decaf::internal::net::tcp::TcpSocket, 3687
  - decaf::net::Socket, 3454
  - decaf::net::SocketImpl, 3477
- getLocalHost
  - decaf::net::InetAddress, 1978
- getLocalPort
  - decaf::net::ServerSocket, 3298
  - decaf::net::Socket, 3454
  - decaf::net::SocketImpl, 3477
- getLogger
  - decaf::util::logging::Logger, 2352
  - decaf::util::logging::LogManager, 2367
- getLoggerName
  - decaf::util::logging::LogRecord, 2372
- getLoggerNames
  - decaf::util::logging::LogManager, 2367
- getLogManager
  - decaf::util::logging::LogManager, 2367
- getLong
  - activemq::commands::ActiveMQMapMessage, 337
  - activemq::util::PrimitiveList, 2935
  - activemq::util::PrimitiveMap, 2946
  - activemq::util::PrimitiveValueNode, 2969
  - cms::MapMessage, 2436
- decaf::internal::nio::ByteBuffer, 974, 975
- decaf::internal::util::ByteArrayAdapter, 941
- decaf::nio::ByteBuffer, 1010
- getLongArray
  - decaf::internal::util::ByteArrayAdapter, 941
- getLongAt
  - decaf::internal::util::ByteArrayAdapter, 941
- getLongCapacity
  - decaf::internal::util::ByteArrayAdapter, 942
- getLongProperty
  - activemq::commands::ActiveMQMessageTemplate, 405
  - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2692
  - cms::Message, 2507
- getLoopbackAddress
  - decaf::net::InetAddress, 1979
- getMagic
  - activemq::commands::WireFormatInfo, 3916
- getManaged
  - decaf::internal::util::GenericResource, 1938
- getMap
  - activemq::commands::ActiveMQMapMessage, 338
  - activemq::util::PrimitiveValueNode, 2969
- getMapNames
  - activemq::commands::ActiveMQMapMessage, 338
  - cms::MapMessage, 2437
- getMarshaledForm
  - activemq::commands::BaseDataStructure, 795
- getMarshaledProperties
  - activemq::wireformat::MarshalAware, 2445
  - activemq::commands::Message, 2483
  - activemq::commands::WireFormatInfo, 3916
- getMaxCacheSize
  - activemq::state::ConnectionStateTracker, 1363
  - activemq::transport::failover::FailoverTransport, 1838
- getMaximumPendingMessageLimit

- activemq::commands::ConsumerInfo, 1430
- getMaximumRedeliveries
  - activemq::core::policies::DefaultRedeliveryPolicy, 1646
  - activemq::core::RedeliveryPolicy, 3124
- getMaxInactivityDuration
  - activemq::commands::WireFormatInfo, 3916
  - activemq::wireformat::openwire::OpenWireFormat, 2841
- getMaxInactivityDurationInitialDelay
  - activemq::commands::WireFormatInfo, 3917
- getMaxInactivityDurationInitialDelay
  - activemq::wireformat::openwire::OpenWireFormat, 2841
- getMaxPrefetchLimit
  - activemq::core::policies::DefaultPrefetchPolicy, 1641
  - activemq::core::PrefetchPolicy, 2927
- getMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1838
- getMaxReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1839
- getMaxThreads
  - decaf::util::concurrent::ThreadPool, 3721
- getMessage
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3121
  - activemq::commands::BrokerError, 825
  - activemq::commands::JournalTrace, 2174
  - activemq::commands::MessageDispatch, 2557
  - activemq::core::DispatchData, 1749
  - cms::CMSException, 1132
  - decaf::lang::Exception, 1798
  - decaf::lang::Throwable, 3726
  - decaf::util::logging::LogRecord, 2372
- getMessageAck
  - activemq::commands::JournalQueueAck, 2118
- getMessageAvailableCount
  - activemq::core::ActiveMQConsumer, 288
- getMessageCount
  - activemq::commands::MessageAck, 2524
- getMessageId
  - activemq::commands::JournalTopicAck, 2146
  - activemq::commands::Message, 2484
  - activemq::commands::MessageDispatchNotification, 2593
  - activemq::commands::MessagePull, 2698
  - getMessageListener
    - activemq::cmsutil::CachedConsumer, 1042
  - activemq::core::ActiveMQConsumer, 288
  - cms::MessageConsumer, 2551
  - getMessageProperties
    - activemq::commands::Message, 2484
  - getMessageSelector
    - activemq::cmsutil::CachedConsumer, 1042
    - activemq::core::ActiveMQConsumer, 288
    - activemq::core::ActiveMQQueueBrowser, 459
    - cms::MessageConsumer, 2552
    - cms::QueueBrowser, 3099
  - getMessageSequenced
    - activemq::commands::JournalTopicAck, 2146
  - getMetaData
    - activemq::core::ActiveMQConnection, 253
    - cms::Connection, 1235
  - getMimeType
    - activemq::commands::ActiveMQBlobMessage, 175
  - getMostSignificantBits
    - decaf::util::UUID, 3903
  - getName
    - activemq::commands::ActiveMQBlobMessage, 175
    - decaf::lang::Thread, 3711
    - decaf::security::auth::x500::X500Principal, 3957
    - decaf::security::Principal, 2975
    - decaf::util::logging::Level, 2292
    - decaf::util::logging::Logger, 2352
  - getNeedClientAuth
    - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796
    - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2800
    - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2815
    - decaf::net::ssl::SSLParameters, 3496

- decaf::net::ssl::SSLServerSocket, 3502
- decaf::net::ssl::SSLSocket, 3510
- getNetworkBrokerId
  - activemq::commands::NetworkBridgeFilter, 2748
- getNetworkConsumerPath
  - activemq::commands::ConsumerInfo, 1430
- getNetworkProperties
  - activemq::commands::BrokerInfo, 859
- getNetworkRuntime
  - decaf::internal::net::Network, 2745
- getNetworkTTL
  - activemq::commands::NetworkBridgeFilter, 2748
- getNextConsumerId
  - activemq::core::ActiveMQSession, 497
- getNextLocalTransactionId
  - activemq::core::ActiveMQConnection, 254
- getNextProducerId
  - activemq::core::ActiveMQSession, 497
- getNextSequenceld
  - activemq::util::LongSequenceGenerator, 2416
- getNextSessionId
  - activemq::core::ActiveMQConnection, 254
- getNextTempDestinationId
  - activemq::core::ActiveMQConnection, 254
- getNotAfter
  - decaf::security::cert::X509Certificate, 3959
- getNotBefore
  - decaf::security::cert::X509Certificate, 3959
- getNumReceivedMessageBeforeFail
  - activemq::transport::mock::MockTransport, 2727
- getNumReceivedMessages
  - activemq::transport::mock::MockTransport, 2727
- getNumSentKeepAlives
  - activemq::transport::mock::MockTransport, 2727
- getNumSentKeepAlivesBeforeFail
  - activemq::transport::mock::MockTransport, 2727
- getNumSentMessageBeforeFail
  - activemq::transport::mock::MockTransport, 2727
- activemq::transport::mock::MockTransport, 2727
- getNumSentMessages
  - activemq::transport::mock::MockTransport, 2727
- getObjectId
  - activemq::commands::RemoveInfo, 3140
- getOOBInline
  - decaf::net::Socket, 3454
- getOperationType
  - activemq::commands::DestinationInfo, 1694
- getOption
  - decaf::internal::net::tcp::TcpSocket, 3687
  - decaf::net::SocketImpl, 3478
- getOptions
  - activemq::commands::ActiveMQDestination, 299
- getOrderedTarget
  - activemq::commands::ActiveMQDestination, 299
- getOriginalDestination
  - activemq::commands::Message, 2484
- getOriginalTransactionId
  - activemq::commands::Message, 2484
- getOutputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2815
  - decaf::internal::net::tcp::TcpSocket, 3688
  - decaf::net::Socket, 3455
  - decaf::net::SocketImpl, 3478
- getParameters
  - activemq::util::CompositeData, 1192
- getParent
  - decaf::util::logging::Logger, 2352
- getParentId
  - activemq::commands::ConsumerId, 1400
  - activemq::commands::ProducerId, 3017
  - activemq::commands::SessionId, 3323
- getPassword
  - activemq::commands::ConnectionInfo, 1328
  - activemq::core::ActiveMQConnection, 254
  - activemq::core::ActiveMQConnectionFactory, 269
- getPath
  - activemq::util::CompositeData, 1192
  - decaf::internal::net::URIType, 3886
  - decaf::net::URI, 3858

getPeerBrokerInfos  
     activemq::commands::BrokerInfo, 859  
 getPhysicalName  
     activemq::commands::ActiveMQDestination, 299  
 getPooledThreadListener  
     decaf::util::concurrent::PooledThread, 2919  
 getPoolSize  
     decaf::util::concurrent::ThreadPool, 3721  
 getPort  
     decaf::internal::net::URIType, 3886  
     decaf::net::Socket, 3455  
     decaf::net::SocketImpl, 3478  
     decaf::net::URI, 3858  
 getPreferredWireFormatInfo  
     activemq::wireformat::openwire::OpenWireFormat, 2841  
 getPrefetch  
     activemq::commands::ConsumerControl, 1372  
 getPrefetchPolicy  
     activemq::core::ActiveMQConnection, 255  
     activemq::core::ActiveMQConnectionFactory, 270  
 getPrefetchSize  
     activemq::commands::ConsumerInfo, 1430  
 getPreparedResult  
     activemq::state::TransactionState, 3814  
 getPriority  
     activemq::cmsutil::CachedProducer, 1047  
     activemq::cmsutil::CmsTemplate, 1146  
     activemq::commands::ConsumerInfo, 1430  
     activemq::commands::Message, 2484  
     activemq::core::ActiveMQProducer, 443  
     cms::MessageProducer, 2684  
     decaf::lang::Thread, 3711  
 getProducerId  
     activemq::commands::Message, 2484  
     activemq::commands::MessageId, 2626, 2627  
     activemq::commands::ProducerAck, 2986  
     activemq::commands::ProducerInfo, 3045  
     activemq::core::ActiveMQProducer, 444  
 getProducerInfo  
     activemq::core::ActiveMQProducer, 444  
 getProducerSequenceId  
     activemq::commands::MessageId, 2627  
     activemq::state::SessionState, 3379  
     activemq::state::TransactionState, 3814  
     activemq::core::ActiveMQConnection, 255  
     activemq::core::ActiveMQConnectionFactory, 270  
     getProperties  
         activemq::commands::WireFormatInfo, 3917  
         activemq::util::ActiveMQProperties, 450  
         activemq::wireformat::stomp::StompFrame, 3579  
         decaf::lang::System, 3675  
         decaf::util::logging::LogManager, 2368  
     getProperty  
         activemq::util::ActiveMQProperties, 451  
         activemq::wireformat::stomp::StompFrame, 3579  
         cms::CMSProperties, 1136, 1137  
         decaf::lang::System, 3675, 3676  
         decaf::util::logging::LogManager, 2368  
         decaf::util::Properties, 3075  
     getPropertyNames  
         activemq::commands::ActiveMQMessageTemplate, 405  
         cms::Message, 2508  
     getProtocols  
         decaf::net::ssl::SSLParameters, 3497  
     getProviderMajorVersion  
         activemq::core::ActiveMQConnectionMetaData, 277  
         cms::ConnectionMetaData, 1357  
     getProviderMinorVersion  
         activemq::core::ActiveMQConnectionMetaData, 278  
         cms::ConnectionMetaData, 1358  
     getProviderVersion  
         activemq::core::ActiveMQConnectionMetaData, 278  
         cms::ConnectionMetaData, 1358  
     getPublicKey  
         decaf::security::cert::Certificate, 1057  
     getQuery  
         decaf::internal::net::URIType, 3886  
         decaf::net::URI, 3858

---

getQueue  
     activemq::core::ActiveMQQueueBrowser, 459  
     cms::QueueBrowser, 3100  
 getQueueBrowserPrefetch  
     activemq::core::policies::DefaultPrefetchPolicy, 1642  
     activemq::core::PrefetchPolicy, 2927  
 getQueueName  
     activemq::commands::ActiveMQQueue, 456  
     activemq::commands::ActiveMQTempQueue, 577  
     cms::Queue, 3094  
     cms::TemporaryQueue, 3702  
 getQueuePrefetch  
     activemq::core::policies::DefaultPrefetchPolicy, 1642  
     activemq::core::PrefetchPolicy, 2927  
 getRawAuthority  
     decaf::net::URI, 3858  
 getRawFragment  
     decaf::net::URI, 3859  
 getRawPath  
     decaf::net::URI, 3859  
 getRawQuery  
     decaf::net::URI, 3859  
 getRawSchemeSpecificPart  
     decaf::net::URI, 3859  
 getRawUserInfo  
     decaf::net::URI, 3860  
 getReadCheckTime  
     activemq::transport::inactivity::InactivityMonitor, 1966  
 getReason  
     decaf::net::URISyntaxException, 3883  
 getReceiveBufferSize  
     decaf::net::ServerSocket, 3298  
     decaf::net::Socket, 3455  
 getReceiveTimeout  
     activemq::cmsutil::CmsTemplate, 1146  
 getReconnectDelay  
     activemq::transport::failover::FailoverTransport, 1839  
 getReconnectTo  
     activemq::commands::ConnectionControl, 1239  
 getRecoveringPullConsumers  
     activemq::state::ConnectionState, 1360  
 getRedeliveryCounter  
     activemq::commands::Message, 2484  
     activemq::commands::MessageDispatch, 2558  
     getRedeliveryDelay  
         activemq::core::policies::DefaultRedeliveryPolicy, 1646  
         activemq::core::RedeliveryPolicy, 3124  
     getRedeliveryPolicy  
         activemq::core::ActiveMQConnection, 255  
         activemq::core::ActiveMQConnectionFactory, 270  
         activemq::core::ActiveMQConsumer, 288  
     getRemaining  
         decaf::util::zip::Inflater, 1988  
     getRemoteAddress  
         activemq::transport::failover::FailoverTransport, 1839  
         activemq::transport::IOTransport, 2108  
         activemq::transport::mock::MockTransport, 2727  
         activemq::transport::Transport, 3820  
         activemq::transport::TransportFilter, 3830  
     getRemoteBlobUrl  
         activemq::commands::ActiveMQBlobMessage, 175  
     getReplyTo  
         activemq::commands::Message, 2485  
     getResourceLifecycleManager  
         activemq::cmsutil::CmsAccessor, 1126  
         activemq::cmsutil::SessionPool, 3377  
     getResponse  
         activemq::transport::correlator::FutureResponse, 1933  
     getResult  
         activemq::commands::IntegerResponse, 2056  
     getReuseAddress  
         decaf::net::ServerSocket, 3298  
         decaf::net::Socket, 3456  
     getRuntime  
         decaf::lang::Runtime, 3266  
     getRuntimeLock  
         decaf::internal::net::Network, 2745  
     getSafeValue  
         decaf::util::StlQueue, 3560  
     getScheme  
         activemq::util::CompositeData, 1192  
         decaf::internal::net::URIType, 3886  
         decaf::net::URI, 3860

---

---

getSchemeSpecificPart	decaf::nio::ByteBuffer, 1011
decaf::internal::net::URIType, 3887	getShortArray
decaf::net::URI, 3860	decaf::internal::util::ByteArrayAdapter,
getSeedFromId	943
activemq::util::IdGenerator, 1952	getShortAt
getSelector	decaf::internal::util::ByteArrayAdapter,
activemq::commands::ConsumerInfo,	943
1430	getShortCapacity
activemq::commands::SubscriptionInfo,	decaf::internal::util::ByteArrayAdapter,
3618, 3619	943
getSendBufferSize	getShortProperty
decaf::net::Socket, 3456	activemq::commands::ActiveMQMessageTemplate,
getSendTimeout	406
activemq::core::ActiveMQConnection,	activemq::wireformat::openwire::utils::MessagePropertyInterce
255	2693
activemq::core::ActiveMQConnectionFactory	cms::Message, 2508
270	getSigAlgName
activemq::core::ActiveMQProducer, 444	decaf::security::cert::X509Certificate,
getSequenceFromId	3959
activemq::util::IdGenerator, 1952	getSigAlgOID
getServerSocketFactory	decaf::security::cert::X509Certificate,
decaf::net::ssl::SSLContext, 3491	3959
getServiceName	getSigAlgParams
activemq::commands::DiscoveryEvent,	decaf::security::cert::X509Certificate,
1724	3959
getSession	getSignature
activemq::cmsutil::PooledSession, 2916	decaf::security::cert::X509Certificate,
getSessionAcknowledgeMode	3959
activemq::cmsutil::CmsAccessor, 1126	getSize
getSessionId	activemq::commands::ActiveMQTextMessage,
activemq::commands::ConsumerId, 1400	634
activemq::commands::ProducerId, 3017	activemq::commands::Message, 2485
activemq::commands::SessionInfo, 3350	activemq::commands::ProducerAck, 2986
activemq::core::ActiveMQSession, 497	getSocketFactory
getSessionInfo	decaf::net::ssl::SSLContext, 3491
activemq::core::ActiveMQSession, 498	getSocketHandle
getSessionState	decaf::internal::net::tcp::TcpSocket, 3688
activemq::state::ConnectionState, 1360	getSoLinger
getSessionStates	decaf::net::Socket, 3456
activemq::state::ConnectionState, 1360	getSoTimeout
getShort	decaf::net::ServerSocket, 3299
activemq::commands::ActiveMQMapMessage	decaf::net::Socket, 3456
338	getSource
activemq::util::PrimitiveList, 2935	decaf::internal::net::URIType, 3887
activemq::util::PrimitiveMap, 2947	getSourceFile
activemq::util::PrimitiveValueNode, 2970	decaf::util::logging::LogRecord, 2372
cms::MapMessage, 2437	getSourceFunction
decaf::internal::nio::ByteBuffer, 975	decaf::util::logging::LogRecord, 2372
decaf::internal::util::ByteArrayAdapter,	getSourceLine
942	decaf::util::logging::LogRecord, 2372

---

- getSSLParameters
  - decaf::net::ssl::SSLSocket, 3511
- getStackTrace
  - cms::CMSException, 1132
  - decaf::lang::Exception, 1798
  - decaf::lang::Throwable, 3726
- getStackTraceElements
  - activemq::commands::BrokerError, 826
- getStackTraceString
  - cms::CMSException, 1132
  - decaf::lang::Exception, 1798
  - decaf::lang::Throwable, 3727
- getStartupMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1839
- getState
  - decaf::lang::Thread, 3712
- getString
  - activemq::commands::ActiveMQMapMessage, 339
  - activemq::util::PrimitiveList, 2936
  - activemq::util::PrimitiveMap, 2947
  - activemq::util::PrimitiveValueNode, 2970
  - cms::MapMessage, 2438
- getStringProperty
  - activemq::commands::ActiveMQMessageTemplate, 406
  - activemq::wireformat::openwire::utils::MessageProperty, 2693
  - cms::Message, 2509
- getSubscriptionName
  - activemq::commands::RemoveSubscriptionInfo, 3168
  - activemq::commands::SubscriptionInfo, 3619
- getSubjectUniqueID
  - decaf::security::cert::X509Certificate, 3960
- getSubjectX500Principal
  - decaf::security::cert::X509Certificate, 3960
- getSubscribedDestination
  - activemq::commands::SubscriptionInfo, 3619
- getSubscriptionName
  - activemq::commands::ConsumerInfo, 1430
- getSubscriptionName
  - activemq::commands::JournalTopicAck, 2146
- getSupportedCipherSuites
  - decaf::internal::net::ssl::DefaultSSLServerSocketFactory, 1662
  - decaf::internal::net::ssl::DefaultSSLSocketFactory, 1669
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2800
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2807
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2816
  - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2831
  - decaf::net::ssl::SSLServerSocket, 3502
  - decaf::net::ssl::SSLServerSocketFactory, 3506
- getSupportedProtocols
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796
  - decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2801
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2816
  - decaf::internal::net::ssl::openssl::SSLServerSocket, 3502
  - decaf::net::ssl::SSLSocket, 3511
- getSupportedSSLParameters
  - decaf::net::ssl::SSLContext, 3491
- getTail
  - decaf::util::logging::Formatter, 1929
  - decaf::util::logging::XMLFormatter, 3990
- getTargetConsumerId
  - activemq::commands::Message, 2485
- getTBSCertificate
  - decaf::security::cert::X509Certificate, 3960
- getTcpNoDelay
  - decaf::net::Socket, 3457
- getTempDesinations
  - activemq::state::ConnectionState, 1360
- getText
  - activemq::commands::ActiveMQTextMessage, 634
  - cms::TextMessage, 3705
- getThrown
  - decaf::util::logging::LogRecord, 2372
- getTime
  -

decaf::util::Date, 1635  
 getTimeout  
     activemq::commands::DestinationInfo, 1694  
     activemq::commands::MessagePull, 2698  
     activemq::transport::failover::FailoverTransport, 1839  
 getTimestamp  
     activemq::commands::Message, 2485  
     decaf::util::logging::LogRecord, 2373  
 getTimeToLive  
     activemq::cmsutil::CachedProducer, 1047  
     activemq::cmsutil::CmsTemplate, 1146  
     activemq::core::ActiveMQProducer, 4449  
     cms::MessageProducer, 2684  
 getTopicName  
     activemq::commands::ActiveMQTempTopic, 606  
     activemq::commands::ActiveMQTopic, 663  
     cms::TemporaryTopic, 3704  
     cms::Topic, 3758  
 getTopicPrefetch  
     activemq::core::policies::DefaultPrefetchPolicy, 1642  
     activemq::core::PrefetchPolicy, 2927  
 getTrafficClass  
     decaf::net::Socket, 3457  
 getTransactionContext  
     activemq::core::ActiveMQSession, 498  
 getTransactionId  
     activemq::commands::JournalTopicAck, 2146  
     activemq::commands::JournalTransaction, 2200  
     activemq::commands::Message, 2485  
     activemq::commands::MessageAck, 2524  
     activemq::commands::TransactionInfo, 3787  
     activemq::core::ActiveMQTransactionContext, 690  
 getTransactionState  
     activemq::state::ConnectionState, 1360  
     activemq::state::ProducerState, 3072  
 getTransactionStates  
     activemq::state::ConnectionState, 1360  
 getTransport  
     activemq::core::ActiveMQConnection, 255  
     activemq::transport::failover::BackupTransport, 719  
     activemq::transport::failover::FailoverTransport, 1839  
     activemq::transport::IOTransport, 2108  
     activemq::transport::mock::MockTransport, 2727  
     activemq::transport::Transport, 3821  
     activemq::transport::TransportFilter, 3830  
 getTransportNames  
     activemq::transport::TransportRegistry, 3839  
 getTreadId  
     decaf::util::logging::LogRecord, 2373  
 getType  
     activemq::commands::JournalTransaction, 2200  
     activemq::commands::Message, 2485  
     activemq::commands::TransactionInfo, 3788  
     activemq::util::PrimitiveValueNode, 2970  
     decaf::security::cert::Certificate, 1057  
 getUncaughtExceptionHandler  
     decaf::lang::Thread, 3712  
 getUnconsumedMessages  
     activemq::core::ActiveMQSessionExecutor, 505  
 getURI  
     activemq::transport::failover::URIPool, 3876  
 getUri  
     activemq::transport::failover::BackupTransport, 719  
 getUsage  
     activemq::util::MemoryUsage, 2474  
 getClientMode  
     decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796  
     decaf::internal::net::ssl::openssl::OpenSSLSocket, 2816  
     decaf::net::ssl::SSLSocket, 3511  
 getParentHandlers  
     decaf::util::logging::Logger, 2353  
 getUserID  
     activemq::commands::Message, 2485  
 getUserInfo  
     decaf::internal::net::URIType, 3887  
     decaf::net::URI, 3860  
 getUsername



activemq::commands::ConnectionInfo, getWireFormat  
 1328  
 activemq::transport::mock::MockTransport,  
 getUsername 2728  
 activemq::core::ActiveMQConnection, getWireFormatNames  
 256  
 activemq::wireformat::WireFormatRegistry,  
 activemq::core::ActiveMQConnectionFactory, 3949  
 271  
 getWriteCheckTime  
 activemq::transport::inactivity::InactivityMonitor,  
 getValue  
 activemq::commands::BrokerId, 831 1966  
 activemq::commands::ConnectionId, 1299  
 GlobalTransactionId  
 1300  
 activemq::commands::XATransactionId,  
 activemq::commands::ConsumerId, 1401 3964  
 activemq::commands::LocalTransactionId  
 2309  
 internal\_state, 2082  
 activemq::commands::ProducerId, 3017  
 groupID  
 activemq::commands::SessionId, 3323  
 activemq::commands::Message, 2491  
 activemq::util::PrimitiveValueNode, 297  
 groupSequence  
 decaf::internal::net::SocketFileDescriptor, activemq::commands::Message, 2491  
 3472  
 GT\_OFF  
 decaf::util::Map::Entry, 1789  
 gzguts.h, 4422  
 decaf::util::zip::Adler32, 692  
 GUNZIP  
 decaf::util::zip::Checksum, 1115  
 inflate.h, 4424  
 decaf::util::zip::CRC32, 1491  
 GZ\_APPEND  
 gzguts.h, 4422  
 getVersion  
 activemq::commands::WireFormatInfo, gz\_header  
 3917  
 zlib.h, 4435  
 activemq::wireformat::openwire::OpenWireFormat, 1938  
 2842  
 comm\_max, 1938  
 activemq::wireformat::stomp::StompWireFormat, 1938  
 3587  
 done, 1939  
 activemq::wireformat::WireFormat, 3909  
 extra, 1939  
 decaf::security::cert::X509Certificate,  
 extra\_len, 1939  
 3960  
 extra\_max, 1939  
 getWantClientAuth  
 hcrc, 1939  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 1939  
 2796  
 name\_max, 1939  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 1939  
 2801  
 text, 1939  
 decaf::internal::net::ssl::openssl::OpenSSLSocket, 1939  
 2816  
 xflags, 1939  
 decaf::net::ssl::SSLParameters, 3497  
 gz\_header  
 decaf::net::ssl::SSLServerSocket, 3502  
 zlib.h, 4435  
 decaf::net::ssl::SSLSocket, 3512  
 GZ\_NONE  
 getWasPrepared  
 gzguts.h, 4422  
 activemq::commands::JournalTransactionId, 2200  
 GZ\_READ  
 gzguts.h, 4422  
 getWhen  
 gz\_state, 1939  
 decaf::util::TimerTask, 3744  
 direct, 1940  
 getWindowSize  
 eof, 1940  
 activemq::commands::ProducerInfo, 3046  
 err, 1940

- fd, 1940
- have, 1940
- how, 1940
- in, 1940
- level, 1940
- mode, 1940
- msg, 1940
- next, 1940
- out, 1940
- path, 1940
- pos, 1940
- raw, 1940
- seek, 1940
- size, 1940
- skip, 1940
- start, 1941
- strategy, 1941
- strm, 1941
- want, 1941
- gz\_statep
  - gzguts.h, 4423
- GZ\_WRITE
  - gzguts.h, 4422
- GZBUFSIZE
  - gzguts.h, 4422
- gzFile
  - zlib.h, 4435
- gzguts.h
  - COPY, 4422
  - GT\_OFF, 4422
  - GZ\_APPEND, 4422
  - GZ\_NONE, 4422
  - GZ\_READ, 4422
  - gz\_statep, 4423
  - GZ\_WRITE, 4422
  - GZBUFSIZE, 4422
  - GZIP, 4422
  - local, 4422
  - LOOK, 4422
  - OF, 4423
  - ZLIB\_INTERNAL, 4422
  - zstrerror, 4422
- gzhead
  - internal\_state, 2083
- gzindex
  - internal\_state, 2083
- GZIP
  - deflate.h, 4420
  - gzguts.h, 4422
- Handler
  - decaf::util::logging::Handler, 1942
- handleTransportFailure
  - activemq::transport::failover::FailoverTransport, 1839
- hasArray
  - decaf::internal::nio::ByteBuffer, 976
  - decaf::internal::nio::CharArrayBuffer, 1086
  - decaf::internal::nio::DoubleArrayBuffer, 1771
  - decaf::internal::nio::FloatArrayBuffer, 1885
  - decaf::internal::nio::IntArrayBuffer, 2024
  - decaf::internal::nio::LongArrayBuffer, 2401
  - decaf::internal::nio::ShortArrayBuffer, 3398
  - decaf::nio::ByteBuffer, 1012
  - decaf::nio::CharBuffer, 1099
  - decaf::nio::DoubleBuffer, 1781
  - decaf::nio::FloatBuffer, 1894
  - decaf::nio::IntBuffer, 2033
  - decaf::nio::LongBuffer, 2410
  - decaf::nio::ShortBuffer, 3408
- hash\_bits
  - internal\_state, 2083
- hash\_mask
  - internal\_state, 2083
- hash\_shift
  - internal\_state, 2083
- hash\_size
  - internal\_state, 2083
- hashCode
  - decaf::security::auth::x500::X500Principal, 3958
- hasMoreMessages
  - activemq::core::ActiveMQQueueBrowser, 459
  - cms::MessageEnumeration, 2620
- hasMoreTokens
  - decaf::util::StringTokenizer, 3614
- hasNegotiator
  - activemq::wireformat::openwire::OpenWireFormat, 2842
  - activemq::wireformat::stomp::StompWireFormat, 3587
  - activemq::wireformat::WireFormat, 3909
- hasNext
  - decaf::util::Iterator, 2115
- hasPrevious
  - decaf::util::ListIterator, 2305
- hasProperty

activemq::util::ActiveMQProperties, 451  
 activemq::wireformat::stomp::StompFrame, 3580  
 cms::CMSProperties, 1137  
 decaf::util::Properties, 3076  
 hasRemaining  
   decaf::nio::Buffer, 890  
 hasUnconsumedMessages  
   activemq::core::ActiveMQSessionExecutor, 505  
 have  
   gz\_state, 1940  
   inflate\_state, 1984  
 HAVE\_PTHREAD\_H  
   activemq/util/Config.h, 4086  
   decaf/util/Config.h, 4087  
 HAVE\_UUID\_T  
   activemq/util/Config.h, 4086  
   decaf/util/Config.h, 4087  
 HAVE\_UUID\_UUID\_H  
   activemq/util/Config.h, 4086  
   decaf/util/Config.h, 4087  
 havedict  
   inflate\_state, 1984  
 HCRC  
   inflate.h, 4424  
 hcrc  
   gz\_header\_s, 1939  
 HCRC\_STATE  
   deflate.h, 4420  
 HEAD  
   inflate.h, 4424  
 head  
   inflate\_state, 1984  
   internal\_state, 2083  
 HEADER\_ACK  
   activemq::wireformat::stomp::StompCommandConstants, 3573  
 HEADER\_CLIENT\_ID  
   activemq::wireformat::stomp::StompCommandConstants, 3573  
 HEADER\_CONSUMERPRIORITY  
   activemq::wireformat::stomp::StompCommandConstants, 3573  
 HEADER\_CONTENTLENGTH  
   activemq::wireformat::stomp::StompCommandConstants, 3573  
 HEADER\_CORRELATIONID  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_DESTINATION  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_DISPATCH\_ASYNC  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_EXCLUSIVE  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_EXPIRES  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_ID  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_JMSPRIORITY  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_LOGIN  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_MAXPENDINGMSGLIMIT  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_MESSAGE  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_MESSAGEID  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_NOLOCAL  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_OLDSUBSCRIPTIONNAME  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_PASSWORD  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_PERSISTENT  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_PREFETCHSIZE  
   activemq::wireformat::stomp::StompCommandConstants, 3574  
 HEADER\_RECEIPT\_REQUIRED  
   activemq::wireformat::stomp::StompCommandConstants, 3575  
 HEADER\_RECEIPTID

activemq::wireformat::stomp::StompCommandConstants, 2083  
 3575 heap\_max  
 HEADER\_REDELIVERED internal\_state, 2083  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 deflate.h, 4420  
 HEADER\_REDELIVERYCOUNT HexStringParser  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 HexTable  
 HEADER\_REPLYTO activemq::wireformat::openwire::utils::HexTable, 3575  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 high\_water  
 HEADER\_REQUESTID internal\_state, 2083  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 decaf::lang::Integer, 2044  
 HEADER\_RESPONSEID decaf::lang::Long, 2383  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 inflate\_state, 1984  
 HEADER\_RETROACTIVE hostname  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 INET\_ADDRESS, 1982  
 HOURS  
 HEADER\_SELECTOR decaf::util::concurrent::TimeUnit, 3757  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 gz\_state, 1940  
 HEADER\_SESSIONID HttpRetryException  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 HttpRetryException, 1949, 1950  
 HEADER\_SUBSCRIPTION HUFFMAN\_ONLY  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 Inflater, 1681  
 HEADER\_SUBSCRIPTIONNAME ID\_ACTIVEMQBLOBMESSAGE  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 activemq::commands::ActiveMQBlobMessage, 177  
 HEADER\_TIMESTAMP ID\_ACTIVEMQBYTESMESSAGE  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 activemq::commands::ActiveMQBytesMessage, 220  
 HEADER\_TRANSACTIONID ID\_ACTIVEMQDESTINATION  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 activemq::commands::ActiveMQDestination, 303  
 HEADER\_TRANSFORMATION ID\_ACTIVEMQMAPMESSAGE  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 activemq::commands::ActiveMQMapMessage, 344  
 HEADER\_TRANSFORMATION\_ERROR ID\_ACTIVEMQMESSAGE  
 activemq::wireformat::stomp::StompCommandConstants, 3575  
 3575 activemq::commands::ActiveMQMessage, 371  
 HEADER\_TYPE ID\_ACTIVEMQOBJECTMESSAGE  
 activemq::wireformat::stomp::StompCommandConstants, 3576  
 3576 activemq::commands::ActiveMQObjectMessage, 416  
 heap ID\_ACTIVEMQQUEUE  
 internal\_state, 2083 activemq::commands::ActiveMQQueue, 457  
 heap\_len 457

ID_ACTIVEMQSTREAMMESSAGE	ID_DESTINATIONINFO
activemq::commands::ActiveMQStreamMessage, 523	activemq::commands::DestinationInfo, 1696
ID_ACTIVEMQTEMPDESTINATION	ID_DISCOVERYEVENT
activemq::commands::ActiveMQTempDestination, 550	activemq::commands::DiscoveryEvent, 1724
ID_ACTIVEMQTEMPQUEUE	ID_EXCEPTIONRESPONSE
activemq::commands::ActiveMQTempQueue, 578	activemq::commands::ExceptionResponse, 1804
ID_ACTIVEMQTEMPTOPIC	ID_FLUSHCOMMAND
activemq::commands::ActiveMQTempTopic, 606	activemq::commands::FlushCommand, 1903
ID_ACTIVEMQTEXTMESSAGE	ID_INTEGERRESPONSE
activemq::commands::ActiveMQTextMessage, 635	activemq::commands::IntegerResponse, 2056
ID_ACTIVEMQTOPIC	ID_JOURNALQUEUEACK
activemq::commands::ActiveMQTopic, 664	activemq::commands::JournalQueueAck, 2119
ID_BROKERID	ID_JOURNALTOPICACK
activemq::commands::BrokerId, 832	activemq::commands::JournalTopicAck, 2147
ID_BROKERINFO	ID_JOURNALTRACE
activemq::commands::BrokerInfo, 862	activemq::commands::JournalTrace, 2174
ID_CONNECTIONCONTROL	ID_JOURNALTRANSACTION
activemq::commands::ConnectionControl, 1241	activemq::commands::JournalTransaction, 2201
ID_CONNECTIONERROR	ID_KEEPLIVEINFO
activemq::commands::ConnectionError, 1269	activemq::commands::KeepAliveInfo, 2228
ID_CONNECTIONID	ID_LASTPARTIALCOMMAND
activemq::commands::ConnectionId, 1300	activemq::commands::LastPartialCommand, 2262
ID_CONNECTIONINFO	ID_LOCALTRANSACTIONID
activemq::commands::ConnectionInfo, 1330	activemq::commands::LocalTransactionId, 2310
ID_CONSUMERCONTROL	ID_MESSAGE
activemq::commands::ConsumerControl, 1373	activemq::commands::Message, 2491
ID_CONSUMERID	ID_MESSAGEACK
activemq::commands::ConsumerId, 1401	activemq::commands::MessageAck, 2526
ID_CONSUMERINFO	ID_MESSAGEDISPATCH
activemq::commands::ConsumerInfo, 1433	activemq::commands::MessageDispatch, 2559
ID_CONTROLCOMMAND	ID_MESSAGEDISPATCHNOTIFICATION
activemq::commands::ControlCommand, 1462	activemq::commands::MessageDispatchNotification, 2595
ID_DATAARRAYRESPONSE	ID_MESSAGEID
activemq::commands::DataArrayResponse, 1496	activemq::commands::MessageId, 2627
ID_DATARESPONSE	ID_MESSAGEPULL
activemq::commands::DataResponse, 1552	activemq::commands::MessagePull, 2699
	ID_NETWORKBRIDGEFILTER

activemq::commands::NetworkBridgeFilter, 2749  
 ID\_PARTIALCOMMAND  
 activemq::commands::PartialCommandIllegalThreadStateException, 2869  
 ID\_PRODUCERACK  
 activemq::commands::ProducerAck, 2987  
 ID\_PRODUCERID  
 activemq::commands::ProducerId, 3018  
 ID\_PRODUCERINFO  
 activemq::commands::ProducerInfo, 3047  
 ID\_REMOVEINFO  
 activemq::commands::RemoveInfo, 3141  
 ID\_REMOVESUBSCRIPTIONINFO  
 InactivityMonitor  
 activemq::commands::RemoveSubscriptionInfo, 3169  
 ID\_REPLAYCOMMAND  
 increaseUsage  
 activemq::commands::ReplayCommand, 3197  
 ID\_RESPONSE  
 incrementAndGet  
 activemq::commands::Response, 3231  
 ID\_SESSIONID  
 indexOf  
 activemq::commands::SessionId, 3324  
 ID\_SESSIONINFO  
 decaf::util::List, 2299  
 activemq::commands::SessionInfo, 3351  
 ID\_SHUTDOWNINFO  
 IndexOutOfBoundsException  
 activemq::commands::ShutdownInfo, 3416  
 ID\_SUBSCRIPTIONINFO  
 decaf::lang::exceptions::IndexOutOfBoundsException, 1968, 1969  
 activemq::commands::SubscriptionInfo, 3620  
 INDIVIDUAL\_ACKNOWLEDGE  
 cms::Session, 3308  
 ID\_TRANSACTIONID  
 InetAddress  
 activemq::commands::TransactionId, 3762  
 ID\_TRANSACTIONINFO  
 decaf::net::Inet4Address, 1971  
 Inet6Address  
 activemq::commands::TransactionInfo, 3789  
 decaf::net::Inet6Address, 1974  
 ID\_WIREFORMATINFO  
 InetSocketAddress  
 decaf::net::Inet4Address, 1973  
 activemq::commands::WireFormatInfo, 3922  
 decaf::net::Inet6Address, 1974  
 decaf::net::InetAddress, 1976  
 ID\_XATransactionID  
 InetSocketAddress  
 activemq::commands::XATransactionId, 3964  
 decaf::net::InetSocketAddress, 1982  
 inflast.h  
 IdGenerator  
 OF, 4423  
 activemq::util::IdGenerator, 1951  
 inflate  
 IllegalArgumentException  
 decaf::util::zip::Inflater, 1989, 1990  
 decaf::lang::exceptions::IllegalArgumentOutOfRangeException, 1953, 1954  
 BAD, 4425  
 IllegalMonitorStateException  
 CHECK, 4425  
 decaf::lang::exceptions::IllegalMonitorStateException, 1956, 1957  
 CODE, 4424  
 COMMENT, 4424  
 IllegalStateException  
 COPY, 4424

- COPY\_, 4424
- DICT, 4424
- DICTID, 4424
- DIST, 4425
- DISTEXT, 4425
- DONE, 4425
- EXLEN, 4424
- EXTRA, 4424
- FLAGS, 4424
- GUNZIP, 4424
- HCRC, 4424
- HEAD, 4424
- inflate\_mode, 4424
- LEN, 4425
- LEN\_, 4424
- LENEXT, 4425
- LENGTH, 4425
- LENLENS, 4424
- LIT, 4425
- MATCH, 4425
- MEM, 4425
- NAME, 4424
- OS, 4424
- STORED, 4424
- SYNC, 4425
- TABLE, 4424
- TIME, 4424
- TYPE, 4424
- TYPEDO, 4424
- inflate\_mode
  - inflate.h, 4424
- inflate\_state, 1982
  - back, 1983
  - bits, 1983
  - check, 1983
  - codes, 1983
  - distbits, 1983
  - distcode, 1984
  - dmax, 1984
  - extra, 1984
  - flags, 1984
  - have, 1984
  - havedict, 1984
  - head, 1984
  - hold, 1984
  - last, 1984
  - lenbits, 1984
  - lencode, 1984
  - length, 1984
  - lens, 1984
  - mode, 1984
  - ncode, 1984
  - ndist, 1984
  - next, 1984
  - nlen, 1984
  - offset, 1984
  - sane, 1984
  - total, 1984
  - was, 1984
  - wbits, 1984
  - whave, 1984
  - window, 1985
  - wnext, 1985
  - work, 1985
  - wrap, 1985
  - wsiz, 1985
- inflateBackInit
  - zlib.h, 4433
- inflateInit
  - zlib.h, 4433
- inflateInit2
  - zlib.h, 4433
- Inflater
  - decaf::util::zip::Inflater, 1987
- inflater
  - decaf::util::zip::InflaterInputStream, 2001
- InflaterInputStream
  - decaf::util::zip::InflaterInputStream, 1997
- INFO
  - decaf::util::logging::Level, 2295
- Info
  - decaf::util::logging, 144
- info
  - decaf::util::logging::Logger, 2353
  - decaf::util::logging::SimpleLogger, 3445
- infrees.h
  - CODES, 4426
  - codetype, 4426
  - DISTS, 4426
  - ENOUGH, 4426
  - ENOUGH\_DISTS, 4426
  - ENOUGH\_LENS, 4426
  - LENS, 4426
  - OF, 4426
- INHERIT
  - decaf::util::logging::Level, 2295
- init
  - activemq::cmsutil::CmsAccessor, 1126
  - activemq::cmsutil::CmsDestinationAccessor, 1129

activemq::cmsutil::CmsTemplate, 1146  
 activemq::cmsutil::DestinationResolver, 1721  
 activemq::cmsutil::DynamicDestinationResolver, 1787  
 INIT\_STATE  
   deflate.h, 4420  
 initCause  
   decaf::lang::Exception, 1799  
   decaf::lang::Throwable, 3727  
 initializeLibrary  
   activemq::library::ActiveMQCPP, 292, 293  
 initializeNetworking  
   decaf::internal::net::Network, 2746  
 initializeRuntime  
   decaf::lang::Runtime, 3266  
 initSocketImpl  
   decaf::net::Socket, 3457  
 inProgressClearRequired  
   activemq::core::ActiveMQConsumer, 289  
 InputStream  
   decaf::io::InputStream, 2004  
 inputStream  
   decaf::io::FilterInputStream, 1860  
 InputStreamReader  
   decaf::io::InputStreamReader, 2014  
 inReceive  
   activemq::wireformat::openwire::OpenWireFormat, 2842  
   activemq::wireformat::stomp::StompWireFormat, 3588  
   activemq::wireformat::WireFormat, 3909  
 ins\_h  
   internal\_state, 2083  
 insert  
   decaf::internal::util::TimerTaskHeap, 3746  
 IntArrayBuffer  
   decaf::internal::nio::IntArrayBuffer, 2019, 2020  
 intBitsToFloat  
   decaf::lang::Float, 1870  
 IntBuffer  
   decaf::nio::IntBuffer, 2028  
 Integer  
   decaf::lang::Integer, 2041  
 INTEGER\_TYPE  
   activemq::util::PrimitiveValueNode, 2963  
 IntegerResponse  
   activemq::commands::IntegerResponse, 2055  
   IntegerResponseMarshaller  
   activemq::wireformat::openwire::marshal::v1::IntegerResponse, 2074  
   activemq::wireformat::openwire::marshal::v2::IntegerResponse, 2062  
   activemq::wireformat::openwire::marshal::v3::IntegerResponse, 2066  
   activemq::wireformat::openwire::marshal::v4::IntegerResponse, 2070  
   activemq::wireformat::openwire::marshal::v5::IntegerResponse, 2078  
   activemq::wireformat::openwire::marshal::v6::IntegerResponse, 2058  
 internal\_state, 2081  
   bi\_buf, 2082  
   bi\_valid, 2082  
   bl\_count, 2082  
   bl\_desc, 2082  
   bl\_tree, 2082  
   block\_start, 2082  
   d\_buf, 2082  
   d\_desc, 2082  
   depth, 2082  
   dummy, 2082  
   dyn\_dtree, 2082  
   dyn\_ltree, 2082  
   good\_match, 2082  
   gzhead, 2083  
   gzindex, 2083  
   hash\_bits, 2083  
   hash\_mask, 2083  
   hash\_shift, 2083  
   hash\_size, 2083  
   head, 2083  
   heap, 2083  
   heap\_len, 2083  
   heap\_max, 2083  
   high\_water, 2083  
   ins\_h, 2083  
   l\_buf, 2083  
   l\_desc, 2083  
   last\_eob\_len, 2083  
   last\_flush, 2083  
   last\_lit, 2083  
   level, 2083  
   lit\_bufsize, 2083  
   lookahead, 2083  
   match\_available, 2083



- match\_length, 2083
- match\_start, 2083
- matches, 2083
- max\_chain\_length, 2084
- max\_lazy\_match, 2084
- method, 2084
- nice\_match, 2084
- opt\_len, 2084
- pending, 2084
- pending\_buf, 2084
- pending\_buf\_size, 2084
- pending\_out, 2084
- prev, 2084
- prev\_length, 2084
- prev\_match, 2084
- static\_len, 2084
- status, 2084
- strategy, 2084
- strm, 2084
- strstart, 2084
- w\_bits, 2084
- w\_mask, 2084
- w\_size, 2084
- window, 2084
- window\_size, 2084
- wrap, 2084
- InternalCommandListener
  - activemq::transport::mock::InternalCommandListener, 2085
- InterruptedException
  - decaf::lang::exceptions::InterruptedException, 2087, 2088
- InterruptedIOException
  - decaf::io::InterruptedIOException, 2090, 2091
- intf
  - zconf.h, 4429
- intValue
  - activemq::util::PrimitiveValueNode::PrimitiveValue, 2958
  - decaf::lang::Byte, 923
  - decaf::lang::Character, 1073
  - decaf::lang::Double, 1756
  - decaf::lang::Float, 1871
  - decaf::lang::Integer, 2044
  - decaf::lang::Long, 2383
  - decaf::lang::Number, 2788
  - decaf::lang::Short, 3384
  - decaf::util::concurrent::atomic::AtomicInteger, 712
- decaf::util::logging::Level, 2293
- InvalidClientIdException
  - cms::InvalidClientIdException, 2092
- InvalidDestinationException
  - cms::InvalidDestinationException, 2093
- InvalidKeyException
  - decaf::security::InvalidKeyException, 2094, 2095
- InvalidMarkException
  - decaf::nio::InvalidMarkException, 2097, 2098
- InvalidSelectorException
  - cms::InvalidSelectorException, 2100
- InvalidStateException
  - decaf::lang::exceptions::InvalidStateException, 2101, 2102
- IOException
  - decaf::io::IOException, 2103, 2104
- IOTransport
  - activemq::transport::IOTransport, 2107
- IPos
  - deflate.h, 4421
- isAbsolute
  - decaf::internal::net::URIType, 3887
  - decaf::net::URI, 3860
- isAdvisory
  - activemq::commands::ActiveMQDestination, 209
- isAlive
  - decaf::lang::Thread, 3712
- isAlwaysSyncSend
  - activemq::core::ActiveMQConnection, 256
  - activemq::core::ActiveMQConnectionFactory, 271
- isAnyLocalAddress
  - decaf::net::Inet4Address, 1971
  - decaf::net::InetAddress, 1979
- isNotAcknowledged
  - activemq::core::ActiveMQSession, 498
- isBackup
  - activemq::transport::failover::FailoverTransport, 1840
- isBound
  - decaf::net::ServerSocket, 3299
  - decaf::net::Socket, 3458
- isBrokerInfo
  - activemq::commands::BaseCommand, 726
  - activemq::commands::BrokerInfo, 860

activemq::commands::Command, 1167  
 isComposite  
 isBrokerMasterConnector  
 activemq::commands::ConnectionInfo, 1328  
 isCompressed  
 isBrowser  
 activemq::commands::ConsumerInfo, 1431  
 isConnected  
 activemq::transport::failover::FailoverTransport, 1840  
 isBusy  
 decaf::util::concurrent::PooledThread, 2919  
 activemq::transport::IOTransport, 2108  
 activemq::transport::mock::MockTransport, 2728  
 isCacheEnabled  
 activemq::commands::WireFormatInfo, 3917  
 activemq::transport::tcp::TcpTransport, 3699  
 activemq::wireformat::openwire::OpenWireFormat, 2842  
 activemq::transport::Transport, 3821  
 activemq::transport::TransportFilter, 3831  
 isCancelled  
 decaf::internal::net::tcp::TcpSocket, 3688  
 decaf::net::Socket, 3458  
 isClientAcknowledged  
 isConnectionAdvisory  
 activemq::core::ActiveMQSession, 498  
 activemq::commands::ActiveMQDestination, 299  
 isClientMaster  
 activemq::commands::ConnectionInfo, 1328  
 isConnectionInfo  
 activemq::commands::BaseCommand, 726  
 isClose  
 activemq::commands::ConnectionControl, 1240  
 activemq::commands::ConsumerControl, 1372  
 isConnectionInterruptProcessingComplete  
 activemq::state::ConnectionState, 1360  
 isClosed  
 activemq::core::ActiveMQConnection, 256  
 isConsumerAdvisory  
 activemq::commands::ActiveMQDestination, 300  
 activemq::core::ActiveMQConsumer, 289  
 isConsumerInfo  
 activemq::core::ActiveMQProducer, 444  
 activemq::core::MessageDispatchChannel, 2562  
 activemq::commands::BaseCommand, 726  
 activemq::transport::failover::BackupTransport, 719  
 activemq::commands::Command, 1167  
 activemq::commands::ConsumerInfo, 1431  
 activemq::transport::failover::FailoverTransport, 1840  
 isDeletedByBroker  
 activemq::transport::IOTransport, 2108  
 activemq::commands::ActiveMQBlobMessage, 176  
 activemq::transport::mock::MockTransport, 2728  
 isDigit  
 activemq::transport::tcp::TcpTransport, 3698  
 decaf::lang::Character, 1073  
 isDispatchAsync  
 activemq::transport::Transport, 3821  
 activemq::commands::ConsumerInfo, 1431  
 activemq::transport::TransportFilter, 3830  
 activemq::commands::ProducerInfo, 3046  
 decaf::internal::net::tcp::TcpSocket, 3688  
 activemq::core::ActiveMQConnection, 256  
 decaf::io::FilterInputStream, 1858  
 decaf::io::FilterOutputStream, 1864  
 decaf::net::ServerSocket, 3299  
 activemq::core::ActiveMQConnectionFactory, 271  
 decaf::net::Socket, 3458

- isDone
  - decaf::util::concurrent::Future, 1932
  - decaf::util::zip::DeflaterOutputStream, 1686
- isDroppable
  - activemq::commands::Message, 2486
- isDuplexConnection
  - activemq::commands::BrokerInfo, 860
- isDupsOkAcknowledge
  - activemq::core::ActiveMQSession, 498
- isEmpty
  - activemq::core::ActiveMQSessionExecutor, 505
  - activemq::core::MessageDispatchChannel, 2562
  - activemq::util::ActiveMQProperties, 451
  - cms::CMSProperties, 1137
  - decaf::internal::util::TimerTaskHeap, 3747
  - decaf::lang::String, 3612
  - decaf::util::AbstractCollection, 154
  - decaf::util::Collection, 1161
  - decaf::util::concurrent::ConcurrentStlMap, 1211
  - decaf::util::concurrent::SynchronousQueue, 3665
  - decaf::util::Map, 2425
  - decaf::util::Properties, 3076
  - decaf::util::StlList, 3539
  - decaf::util::StlMap, 3550
  - decaf::util::StlSet, 3569
- isEnabled
  - activemq::transport::failover::BackupTransport, 722
- isExclusive
  - activemq::commands::ActiveMQDestination, 300
  - activemq::commands::ConsumerInfo, 1431
- isExit
  - activemq::commands::ConnectionControl, 1240
- isExpired
  - activemq::commands::Message, 2486
- isExplicitQosEnabled
  - activemq::cmsutil::CmsTemplate, 1146
- isFailOnClose
  - activemq::transport::mock::MockTransport, 2728
- isFailOnKeepAliveSends
  - activemq::transport::mock::MockTransport, 2729
  - isFailOnReceiveMessage
  - activemq::transport::mock::MockTransport, 2729
  - isFailOnSendMessage
  - activemq::transport::mock::MockTransport, 2729
  - isFailOnStart
  - activemq::transport::mock::MockTransport, 2729
  - isFailOnStop
  - activemq::transport::mock::MockTransport, 2729
  - isFair
  - decaf::util::concurrent::locks::ReentrantLock, 3129
  - decaf::util::concurrent::Semaphore, 3286
  - isFaultTolerant
  - activemq::commands::ConnectionControl, 1240
  - activemq::commands::ConnectionInfo, 1328
  - activemq::transport::failover::FailoverTransport, 1840
  - activemq::transport::IOTransport, 2108
  - activemq::transport::mock::MockTransport, 2729
  - activemq::transport::tcp::TcpTransport, 3699
  - activemq::transport::Transport, 3821
  - activemq::transport::TransportFilter, 3831
  - isFaultTolerantConfiguration
  - activemq::commands::BrokerInfo, 860
  - isFlush
  - activemq::commands::ConsumerControl, 1372
  - isFull
  - activemq::util::MemoryUsage, 2474
  - activemq::util::Usage, 3896
  - isHeldByCurrentThread
  - decaf::util::concurrent::locks::ReentrantLock, 3129
  - isIndividualAcknowledge
  - activemq::core::ActiveMQSession, 498
  - isInfinite
  - decaf::lang::Double, 1756
  - decaf::lang::Float, 1871
  - isInitialized

activemq::transport::failover::FailoverTransport 1840  
 isInputShutdown  
   decaf::net::Socket, 3458  
 isInTransaction  
   activemq::core::ActiveMQTransactionCoordinator 690  
 isISOControl  
   decaf::lang::Character, 1073  
 isKeepAliveInfo  
   activemq::commands::BaseCommand, 726  
   activemq::commands::Command, 1167  
   activemq::commands::KeepAliveInfo, 2227  
 isKeepAliveResponseRequired  
   activemq::transport::inactivity::InactivityMonitor 1966  
 isLetter  
   decaf::lang::Character, 1073  
 isLetterOrDigit  
   decaf::lang::Character, 1074  
 isLinkLocalAddress  
   decaf::net::Inet4Address, 1971  
   decaf::net::InetAddress, 1979  
 isLocked  
   decaf::util::concurrent::Lock, 2335  
   decaf::util::concurrent::locks::ReentrantLock, 3129  
 isLoggable  
   decaf::util::logging::Filter, 1853  
   decaf::util::logging::Handler, 1943  
   decaf::util::logging::Logger, 2353  
   decaf::util::logging::StreamHandler, 3593  
 isLoopbackAddress  
   decaf::net::Inet4Address, 1972  
   decaf::net::InetAddress, 1979  
 isLowerCase  
   decaf::lang::Character, 1074  
 isManageable  
   activemq::commands::ConnectionInfo, 1328  
 isMarshalAware  
   activemq::commands::ActiveMQMapMessage, 339  
   activemq::commands::BaseDataStructure, 796  
   activemq::commands::Message, 2486  
   activemq::commands::WireFormatInfo, 3918  
   activemq::wireformat::MarshalAware, 2486  
 isMasterBroker  
   activemq::commands::BrokerInfo, 860  
 isMCGlobal  
   decaf::net::Inet4Address, 1972  
   decaf::net::InetAddress, 1980  
 isMCLinkLocal  
   decaf::net::Inet4Address, 1972  
   decaf::net::InetAddress, 1980  
 isMCNodeLocal  
   decaf::net::Inet4Address, 1972  
   decaf::net::InetAddress, 1980  
 isMCOrgLocal  
   decaf::net::Inet4Address, 1972  
   decaf::net::InetAddress, 1980  
 isMCSiteLocal  
   decaf::net::Inet4Address, 1972  
   decaf::net::InetAddress, 1980  
 isMessage  
   activemq::commands::BaseCommand, 727  
   activemq::commands::Command, 1167  
   activemq::commands::Message, 2486  
 isMessageAck  
   activemq::commands::BaseCommand, 727  
   activemq::commands::Command, 1167  
   activemq::commands::MessageAck, 2524  
 isMessageDispatch  
   activemq::commands::BaseCommand, 727  
   activemq::commands::Command, 1167  
   activemq::commands::MessageDispatch, 2558  
 isMessageDispatchNotification  
   activemq::commands::BaseCommand, 727  
   activemq::commands::Command, 1167  
   activemq::commands::MessageDispatchNotification, 2593  
 isMessageIdEnabled  
   activemq::cmsutil::CmsTemplate, 1147  
 isMessageTimestampEnabled  
   activemq::cmsutil::CmsTemplate, 1147  
 isMulticastAddress  
   decaf::net::Inet4Address, 1973  
   decaf::net::InetAddress, 1981  
 isNaN  
   decaf::lang::Double, 1757  
   decaf::lang::Float, 1871  
 isNetworkConnection

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

activemq::state::ConnectionStateTracker,	activemq::core::ActiveMQConnection,
1363	256
isRestoreProducers	activemq::core::ActiveMQSession, 499
activemq::state::ConnectionStateTracker	isStop
1363	activemq::commands::ConsumerControl,
isRestoreSessions	1372
activemq::state::ConnectionStateTracker	isSuspend
1363	activemq::commands::ConnectionControl,
isRestoreTransaction	1240
activemq::state::ConnectionStateTracker	isSynchronizationRegistered
1363	activemq::core::ActiveMQConsumer, 289
isResume	isTcpNoDelayEnabled
activemq::commands::ConnectionControl,	activemq::commands::WireFormatInfo,
1240	3918
isRetroactive	activemq::wireformat::openwire::OpenWireFormat,
activemq::commands::ConsumerInfo,	2843
1431	isTemporary
isRunning	activemq::commands::ActiveMQDestination,
activemq::core::ActiveMQSessionExecutor,	300
505	isTightEncodingEnabled
activemq::core::MessageDispatchChannel,	activemq::commands::WireFormatInfo,
2562	3918
isScheduled	activemq::wireformat::openwire::OpenWireFormat,
decaf::util::TimerTask, 3744	2843
isServerAuthority	isTopic
decaf::internal::net::URIType, 3888	activemq::commands::ActiveMQDestination,
isShutdownInfo	301
activemq::commands::BaseCommand, isTrackMessages	
728	activemq::state::ConnectionStateTracker,
activemq::commands::Command, 1168	1363
activemq::commands::ShutdownInfo, 3415	activemq::transport::failover::FailoverTransport,
isSiteLocalAddress	1841
decaf::net::Inet4Address, 1973	isTrackTransactionProducers
decaf::net::InetAddress, 1981	activemq::state::ConnectionStateTracker,
isSizePrefixDisabled	1363
activemq::commands::WireFormatInfo,	activemq::transport::failover::FailoverTransport,
3918	1841
activemq::wireformat::openwire::OpenWireFormat,	isTrackTransactions
2843	activemq::state::ConnectionStateTracker,
isSlaveBroker	1363
activemq::commands::BrokerInfo, 860	isTransacted
isStackTraceEnabled	activemq::cmsutil::PooledSession, 2916
activemq::commands::WireFormatInfo,	activemq::core::ActiveMQSession, 499
3918	cms::Session, 3317
activemq::wireformat::openwire::OpenWireFormat,	isTransactionInfo
2843	activemq::commands::BaseCommand,
isStart	728
activemq::commands::ConsumerControl,	activemq::commands::Command, 1169
1372	activemq::commands::TransactionInfo,
isStarted	3788

- isTransportFailed
  - activemq::core::ActiveMQConnection, 256
- isUpperCase
  - decaf::lang::Character, 1074
- isUseAsyncSend
  - activemq::core::ActiveMQConnection, 257
  - activemq::core::ActiveMQConnectionFactory, 271
- isUseCollisionAvoidance
  - activemq::core::policies::DefaultRedeliveryPolicy, 1646
  - activemq::core::RedeliveryPolicy, 3125
- isUseCompression
  - activemq::core::ActiveMQConnection, 257
  - activemq::core::ActiveMQConnectionFactory, 271
- isUseExponentialBackOff
  - activemq::core::policies::DefaultRedeliveryPolicy, 1646
  - activemq::core::RedeliveryPolicy, 3125
  - activemq::transport::failover::FailoverTransport, 1841
- isValid
  - activemq::commands::WireFormatInfo, 3919
  - decaf::internal::net::URIType, 3888
- isValidDomainName
  - decaf::internal::net::URIHelper, 3869
- isValidHexChar
  - decaf::internal::net::URIHelper, 3870
- isValidHost
  - decaf::internal::net::URIHelper, 3870
- isValidIP4Word
  - decaf::internal::net::URIHelper, 3870
- isValidIP6Address
  - decaf::internal::net::URIHelper, 3871
- isValidIPv4Address
  - decaf::internal::net::URIHelper, 3871
- isWaitingForResponse
  - activemq::state::Tracked, 3759
- isWhitespace
  - decaf::lang::Character, 1074
- isWildcard
  - activemq::commands::ActiveMQDestination, 301
- isWireFormatInfo
  - activemq::commands::BaseCommand, 728
  - activemq::commands::Command, 1169
  - activemq::commands::WireFormatInfo, 3919
- itemExists
  - activemq::commands::ActiveMQMapMessage, 339
  - cms::MapMessage, 2438
- iterate
  - activemq::core::ActiveMQConsumer, 289
  - activemq::core::ActiveMQSessionExecutor, 506
  - activemq::threads::CompositeTaskRunner, 1195
  - activemq::threads::Task, 3679
  - activemq::transport::failover::BackupTransportPool, 722
  - activemq::transport::failover::CloseTransportsTask, 1122
  - activemq::transport::failover::FailoverTransport, 1841
- iterator
  - decaf::lang::Iterable, 2113, 2114
  - decaf::util::concurrent::SynchronousQueue, 3665
  - decaf::util::PriorityQueue, 2980
  - decaf::util::StlList, 3539
  - decaf::util::StlQueue, 3560
  - decaf::util::StlSet, 3570
- join
  - decaf::lang::Thread, 3712, 3713
- JournalQueueAck
  - activemq::commands::JournalQueueAck, 2117
- JournalQueueAckMarshaller
  - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2140
  - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 2124
  - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 2132
  - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 2136
  - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 2128
  - activemq::wireformat::openwire::marshal::v6::JournalQueueAckMarshaller, 2120
- JournalTopicAck
  - activemq::commands::JournalTopicAck, 2120

activemq::commands::JournalTopicAck, 2144  
 JournalTopicAckMarshaller  
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2169  
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 2153  
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 2157  
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 2165  
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 2149  
 activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller, 2161  
 JournalTrace  
 activemq::commands::JournalTrace, 2172  
 JournalTraceMarshaller  
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2191  
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 2175  
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 2179  
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 2187  
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 2195  
 activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller, 2183  
 JournalTransaction  
 activemq::commands::JournalTransaction, 2199  
 JournalTransactionMarshaller  
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2222  
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 2206  
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 2210  
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 2218  
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 2214  
 activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller, 2202  
 KeepAliveInfo  
 activemq::commands::KeepAliveInfo, 2226  
 KeepAliveInfoMarshaller  
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2250  
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 2257  
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 2258  
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 2242  
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 2246  
 activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller, 2255  
 KeyException  
 decaf::security::KeyException, 2255  
 KeyManagementException  
 decaf::security::KeyManagementException, 2258, 2259  
 KeySet  
 decaf::util::concurrent::ConcurrentStlMap, 2171  
 decaf::util::Map, 2426  
 decaf::util::StlMap, 3550  
 \_buf  
 internal\_state, 2083  
 L\_CODES  
 deflate.h, 4420  
 \_desc  
 internal\_state, 2083  
 last  
 inflate\_state, 1984  
 last\_eob\_len  
 internal\_state, 2083  
 last\_flush  
 internal\_state, 2083  
 last\_lit  
 internal\_state, 2083  
 lastDeliveredSequenceId  
 activemq::commands::RemoveInfo, 3141  
 lastIndexOf  
 decaf::util::List, 2300  
 decaf::util::StlList, 3539  
 lastMessageId  
 activemq::commands::MessageAck, 2526  
 lastNakNumber  
 activemq::commands::ReplayCommand, 3197  
 LastPartialCommand



activemq::commands::LastPartialCommand, decaf::util::logging::Level, 2292  
 2261 level  
 LastPartialCommandMarshaller gz\_state, 1940  
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2083  
 2284 Levels  
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 2083  
 2272 decaf::util::logging::Level  
 limit  
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 2083  
 2268 decaf::nio::CharBuffer, 1099  
 LineNumber  
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 2083  
 2280 activemq::commands::BrokerError::StackTraceElement, 3521  
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 2276  
 List  
 activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller, 2263  
 decaf::util::List, 2297  
 List::TYPE  
 activemq::util::PrimitiveValueNode, 2964  
 LEN listen  
 inflate.h, 4425  
 Len decaf::internal::net::tcp::TcpSocket, 3688  
 deflate.h, 4420 decaf::net::SocketImpl, 3478  
 len listener  
 ct\_data\_s, 1493 activemq::transport::TransportFilter, 3835  
 LEN\_ listIterator  
 inflate.h, 4424 decaf::util::List, 2300, 2301  
 lenbits decaf::util::StlList, 3540  
 inflate\_state, 1984 listValue  
 lencode activemq::util::PrimitiveValueNode::PrimitiveValue, 2958  
 inflate\_state, 1984  
 LENEXT LIT  
 inflate.h, 4425 inflate.h, 4425  
 LENGTH lit\_bufsize  
 inflate.h, 4425 internal\_state, 2083  
 length LITERALS  
 decaf::internal::nio::CharArrayBuffer, 1089 deflate.h, 4420  
 decaf::lang::ArrayPointer, 701 load  
 decaf::lang::CharSequence, 1108 decaf::util::Properties, 3076, 3077  
 decaf::lang::String, 3612 local  
 decaf::nio::CharBuffer, 1099 gzguts.h, 4422  
 decaf::util::zip::InflaterInputStream, 2001 zutil.h, 4439  
 inflate\_state, 1984 localPort  
 LENGTH\_CODES decaf::net::SocketImpl, 3481  
 deflate.h, 4420 LocalTransactionId  
 LENLENS activemq::commands::LocalTransactionId, 2308  
 inflate.h, 4424  
 LENS LocalTransactionIdMarshaller  
 inftrees.h, 4426 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2331  
 lens activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 2315  
 inflate\_state, 1984  
 Less activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 2319  
 decaf::util::comparators::Less, 2287  
 Level

activemq::wireformat::openwire::marshal::v4::LoggerDefines.h, 4533  
 2327 LOGDECAF\_FATAL  
 activemq::wireformat::openwire::marshal::v5::LoggerDefines.h, 4533  
 2323 LOGDECAF\_INFO  
 activemq::wireformat::openwire::marshal::v6::LoggerDefines.h, 4533  
 2311 LOGDECAF\_INITIALIZE  
 Lock  
 decaf::util::concurrent::Lock, 2335 LOGDECAF\_WARN  
 lock  
 LoggerDefines.h, 4533  
 activemq::core::MessageDispatchChannel, 2562  
 2562 decaf::util::logging::Logger, 2348  
 decaf::internal::util::concurrent::MutexImpl, 2743  
 2743 LoggerDefines.h  
 decaf::internal::util::concurrent::Synchronization, 3656  
 3656 LOGDECAF\_DEBUG, 4532  
 decaf::io::InputStream, 2005 LOGDECAF\_DEBUG\_1, 4532  
 decaf::io::OutputStream, 2859 LOGDECAF\_DECLARE, 4533  
 decaf::util::AbstractCollection, 154 LOGDECAF\_DECLARE\_LOCAL, 4533  
 decaf::util::concurrent::ConcurrentStlMap, 1211  
 1211 LOGDECAF\_ERROR, 4533  
 decaf::util::concurrent::Lock, 2335 LOGDECAF\_FATAL, 4533  
 decaf::util::concurrent::locks::Lock, 2338 LOGDECAF\_INFO, 4533  
 decaf::util::concurrent::locks::ReentrantLock, 3129  
 3129 LOGDECAF\_INITIALIZE, 4533  
 decaf::util::concurrent::Mutex, 2737 LOGDECAF\_WARN, 4533  
 decaf::util::concurrent::Synchronizable, 3645  
 3645 LoggerHierarchy  
 decaf::util::concurrent::locks::ReentrantLock, 3129  
 3129 LoggingInputStream  
 decaf::util::concurrent::Mutex, 2737 activemq::io::LoggingInputStream, 2358  
 decaf::util::concurrent::Synchronizable, 3645 LoggingOutputStream  
 3645 activemq::io::LoggingOutputStream, 2360  
 decaf::util::StlMap, 3551 LoggingTransport  
 decaf::util::StlQueue, 3560 activemq::transport::logging::LoggingTransport,  
 2361  
 lock\_count  
 decaf::util::concurrent::MutexHandle, 2744 LogManager  
 2744  
 lock\_owner  
 decaf::util::concurrent::MutexHandle, 2744 LogRecord  
 2744  
 lockInterruptibly  
 decaf::util::concurrent::locks::Lock, 2338 decaf::util::logging::LogRecord, 2371  
 decaf::util::concurrent::locks::ReentrantLock, 3130  
 3130 LogWriter  
 decaf::util::concurrent::locks::ReentrantLock, 3130  
 3130 decaf::util::logging::LogWriter, 2376  
 Long  
 log  
 decaf::lang::Long, 2379  
 decaf::util::logging::Logger, 2353, 2354 LONG\_TYPE  
 decaf::util::logging::LogWriter, 2376 activemq::util::PrimitiveValueNode, 2964  
 decaf::util::logging::SimpleLogger, 3445 LongArrayBuffer  
 LOGDECAF\_DEBUG  
 decaf::internal::nio::LongArrayBuffer, 2396,  
 LoggerDefines.h, 4532 2397  
 LOGDECAF\_DEBUG\_1  
 LoggerDefines.h, 4532 longBitsToDouble  
 LOGDECAF\_DECLARE  
 LoggerDefines.h, 4533 decaf::lang::Double, 1757  
 LOGDECAF\_DECLARE\_LOCAL  
 LoggerDefines.h, 4533 LongBuffer  
 LOGDECAF\_ERROR  
 LongSequenceGenerator  
 activemq::util::LongSequenceGenerator,  
 2416

longValue  
     activemq::util::PrimitiveValueNode::PrimitiveValue744  
         2958  
     decaf::lang::Byte, 923  
     decaf::lang::Character, 1074  
     decaf::lang::Double, 1757  
     decaf::lang::Float, 1872  
     decaf::lang::Integer, 2044  
     decaf::lang::Long, 2383  
     decaf::lang::Number, 2788  
     decaf::lang::Short, 3385  
     decaf::util::concurrent::atomic::AtomicInteger, 712  
 LOOK  
     gzguts.h, 4422  
 lookahead  
     internal\_state, 2083  
 loopbackBytes  
     decaf::net::InetAddress, 1982  
 looseMarshal  
     activemq::wireformat::openwire::marshal::BaseDataStructureMarshaller, 776  
     activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1591  
     activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 183  
     activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 226  
     activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 309  
     activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 350  
     activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 376  
     activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 422  
     activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 466  
     activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 529  
     activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 556  
     activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 584  
     activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 616  
     activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 645  
     activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 673  
     activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 744  
     activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 841  
     activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 872  
     activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1251  
     activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1283  
     activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1314  
     activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1344  
     activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1387  
     activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1415  
     activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1448  
     activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1476  
     activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1510  
     activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1575  
     activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1709  
     activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1742  
     activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1826  
     activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1921  
     activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2074  
     activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2141  
     activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2169  
     activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2192  
     activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 2223  
     activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 2250  
     activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 2284  
     activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 2332

activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller,openwire::marshal::v2::ActiveMQDestinationMarshaller 2544 321  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDispatcherMarshaller,openwire::marshal::v2::ActiveMQDispatcherMarshaller 2584 362  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDispatchOptionsMarshaller,openwire::marshal::v2::ActiveMQDispatchOptionsMarshaller 2613 388  
 activemq::wireformat::openwire::marshal::v1::ActiveMQIDMarshaller,openwire::marshal::v2::ActiveMQIDMarshaller 2649 434  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,openwire::marshal::v2::ActiveMQQueueMarshaller 2671 478  
 activemq::wireformat::openwire::marshal::v1::ActiveMQRuleMarshaller,openwire::marshal::v2::ActiveMQRuleMarshaller 2717 541  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBridgeFormatMarshaller,openwire::marshal::v2::ActiveMQBridgeFormatMarshaller 2771 567  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,openwire::marshal::v2::ActiveMQWireFormatMarshaller 2893 596  
 activemq::wireformat::openwire::marshal::v1::ActiveMQAddMarshaller,openwire::marshal::v2::ActiveMQAddMarshaller 3009 624  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDeleteMarshaller,openwire::marshal::v2::ActiveMQDeleteMarshaller 3040 657  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,openwire::marshal::v2::ActiveMQWireFormatMarshaller 3057 685  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireMarshaller,openwire::marshal::v2::ActiveMQWireMarshaller 3155 765  
 activemq::wireformat::openwire::marshal::v1::ActiveMQSubscriptionInfoMarshaller,openwire::marshal::v2::ActiveMQSubscriptionInfoMarshaller 3171 853  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueInfoMarshaller,openwire::marshal::v2::ActiveMQQueueInfoMarshaller 3202 884  
 activemq::wireformat::openwire::marshal::v1::ActiveMQResponseMarshaller,openwire::marshal::v2::ActiveMQResponseMarshaller 3257 1263  
 activemq::wireformat::openwire::marshal::v1::ActiveMQErrorIdMarshaller,openwire::marshal::v2::ActiveMQErrorIdMarshaller 3346 1271  
 activemq::wireformat::openwire::marshal::v1::ActiveMQInfoIdMarshaller,openwire::marshal::v2::ActiveMQInfoIdMarshaller 3361 1302  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,openwire::marshal::v2::ActiveMQWireFormatMarshaller 3425 1332  
 activemq::wireformat::openwire::marshal::v1::ActiveMQSubscriptionInfoMarshaller,openwire::marshal::v2::ActiveMQSubscriptionInfoMarshaller 3626 1375  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,openwire::marshal::v2::ActiveMQWireFormatMarshaller 3767 1403  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,openwire::marshal::v2::ActiveMQWireFormatMarshaller 3795 1436  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,openwire::marshal::v2::ActiveMQWireFormatMarshaller 3940 1464  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueInfoMarshaller,openwire::marshal::v2::ActiveMQQueueInfoMarshaller 3978 1497  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueInfoMarshaller,openwire::marshal::v2::ActiveMQQueueInfoMarshaller 191 1563  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueInfoMarshaller,openwire::marshal::v2::ActiveMQQueueInfoMarshaller 242 1697

activemq::wireformat::openwire::marshal::v2::DiscoveryErrorMarshaller,	1730
activemq::wireformat::openwire::marshal::v2::DiscoveryExceptionResponseMarshaller,	1810
activemq::wireformat::openwire::marshal::v2::ClusterConnectionInfoMarshaller,	1909
activemq::wireformat::openwire::marshal::v2::ClientResponseMarshaller,	2062
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller,	2125
activemq::wireformat::openwire::marshal::v2::ConversationIdMarshaller,	2153
activemq::wireformat::openwire::marshal::v2::ConversationInfoMarshaller,	2176
activemq::wireformat::openwire::marshal::v2::ConversationTransactionalMarshaller,	2207
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	2234
activemq::wireformat::openwire::marshal::v2::CreateTopicInfoMarshaller,	2272
activemq::wireformat::openwire::marshal::v2::CreateTransactionIdMarshaller,	2316
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	2532
activemq::wireformat::openwire::marshal::v2::CreateTopicInfoMarshaller,	2568
activemq::wireformat::openwire::marshal::v2::CreateTransactionalMarshaller,	2601
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	2629
activemq::wireformat::openwire::marshal::v2::CreateTopicInfoMarshaller,	2662
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	2701
activemq::wireformat::openwire::marshal::v2::CreateTopicInfoMarshaller,	2751
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	2876
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	2989
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	3020
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	3053
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	3143
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	3179
activemq::wireformat::openwire::marshal::v2::CreateQueueInfoMarshaller,	3206
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller,	3243
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller,	3326
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller,	3369
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller,	3421
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller,	3642
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller,	3771
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller,	3811
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller,	3932
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller,	3970
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	179
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller,	222
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	305
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller,	346
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller,	372
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	418
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller,	462
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller,	525
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller,	552
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller,	580
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller,	608
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller,	637
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller,	665
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller,	731
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,	833
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller,	864

activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::MessageIdMarshaller 1243  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::MessageMarshaller 1275  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::MessagePullMarshaller 1306  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::NetworkBridgeFilter 1336  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::PartialCommandMarshaller 1379  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::ProducerAckMarshaller 1407  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::ProducerIdMarshaller 1440  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::ProducerInfoMarshaller 1468  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::RemoveInfoMarshaller 1502  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::RemoveSubscriptionMarshaller 1567  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::ReplayCommandMarshaller 1701  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::ResponseMarshaller 1734  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::SessionIdMarshaller 1814  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::SessionInfoMarshaller 1913  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::ShutdownInfoMarshaller 2066  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::SubscriptionInfoMarshaller 2133  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::TransactionIdMarshaller 2157  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::TransactionInfoMarshaller 2180  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::WireFormatInfoMarshaller 2211  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v3::XATransactionIdMarshaller 2238  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v4::ActiveMQBlobMarshaller 2268  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v4::ActiveMQBytesMarshaller 2320  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v4::ActiveMQDestinationMarshaller 2536  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v4::ActiveMQMapMarshaller 2572  
 activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionFactory::marshal::v4::ActiveMQMessageMarshaller 2605

activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller,	2070
426	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueFormatOpenWire::marshal::v4::JournalQueueAckMarshaller,	2137
470	
activemq::wireformat::openwire::marshal::v4::ActiveMQSimpleMessageMarshaller,	2165
533	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller,	2188
560	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller,	2219
588	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller;	2242
612	marshal::v4::KeepAliveInfoMarshaller,
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller,	2280
641	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller,	2328
669	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireIdMarshaller,	2540
738	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::MessageDispatchMarshaller,	2580
837	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::MessageDispatchNotificationM	2609
868	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::MessageIdMarshaller,	2633
1247	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::MessageMarshaller,	2667
1279	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::MessagePullMarshaller,	2713
1310	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,	2767
1340	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::PartialCommandMarshaller,	2889
1383	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::ProducerAckMarshaller,	2993
1411	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::ProducerIdMarshaller,	3024
1444	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::ProducerInfoMarshaller,	3049
1472	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::RemoveInfoMarshaller,	3163
1506	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller,	3191
1571	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::ReplayCommandMarshaller,	3198
1705	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::ResponseMarshaller,	3238
1738	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::SessionIdMarshaller,	3330
1822	
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormat::openwire::marshal::v4::SessionInfoMarshaller,	3373
1917	

activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatOpenwire::marshal::v5::ConnectionInfoMarshaller	3437	1348
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatOpenwire::marshal::v5::ConsumerControlMarshaller	3634	1391
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatOpenwire::marshal::v5::ConsumerIdMarshaller	3779	1419
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatOpenwire::marshal::v5::ConsumerInfoMarshaller	3807	1452
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatOpenwire::marshal::v5::ControlCommandMarshaller	3936	1480
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatOpenwire::marshal::v5::DataArrayResponseMarshaller	3974	1514
activemq::wireformat::openwire::marshal::v5::ActiveMQBinaryMessageMarshaller	195	1554
activemq::wireformat::openwire::marshal::v5::ActiveMQBinaryMessageMarshaller	234	1717
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	317	1746
activemq::wireformat::openwire::marshal::v5::ActiveMQErrorMessageMarshaller	358	1818
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	384	1925
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	430	2078
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	474	2129
activemq::wireformat::openwire::marshal::v5::ActiveMQSerializedMessageMarshaller	537	2149
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	563	2196
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	592	2215
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	620	2246
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	649	2276
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	677	2324
activemq::wireformat::openwire::marshal::v5::BaseQueueWireFormatOpenwire::marshal::v5::MessageAckMarshaller	751	2548
activemq::wireformat::openwire::marshal::v5::BrokerWireFormatOpenwire::marshal::v5::MessageDispatchMarshaller	845	2576
activemq::wireformat::openwire::marshal::v5::BrokerWireFormatOpenwire::marshal::v5::MessageDispatchMarshaller	876	2617
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	1255	2637
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	1287	2654
activemq::wireformat::openwire::marshal::v5::ConnectionWireFormatOpenwire::marshal::v5::MessagePullMarshaller	1318	2705



activemq::wireformat::openwire::marshal::v5::ActiveMQBridgeFinalV4Marshaller,;marshal::v6::ActiveMQTempDestinationMar  
 2759 571  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ActiveMQTempQueueMarshaller,  
 2880 600  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ActiveMQTempTopicMarshaller,  
 3001 628  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ActiveMQTextMessageMarshaller,  
 3032 653  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ActiveMQTopicMarshaller,  
 3061 681  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::BaseCommandMarshaller,  
 3159 758  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::BrokerIdMarshaller,  
 3187 849  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::BrokerInfoMarshaller,  
 3218 880  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ConnectionControlMarshaller,  
 3247 1259  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ConnectionErrorMarshaller,  
 3338 1291  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ConnectionIdMarshaller,  
 3357 1322  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ConnectionInfoMarshaller,  
 3429 1352  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ConsumerControlMarshaller,  
 3630 1395  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ConsumerIdMarshaller,  
 3764 1423  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ConsumerInfoMarshaller,  
 3791 1456  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::ControlCommandMarshaller,  
 3924 1484  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireAndV4Marshaller,;marshal::v6::DataArrayResponseMarshaller,  
 3986 1518  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageMarshaller,;marshal::v6::DataResponseMarshaller,  
 199 1558  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageMarshaller,;marshal::v6::DestinationInfoMarshaller,  
 238 1713  
 activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller,;marshal::v6::DiscoveryEventMarshaller,  
 325 1726  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller,;marshal::v6::ExceptionResponseMarshaller,  
 366 1806  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller,;marshal::v6::FlushCommandMarshaller,  
 392 1905  
 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller,;marshal::v6::IntegerResponseMarshaller,  
 438 2058  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller,;marshal::v6::JournalQueueAckMarshaller,  
 482 2121  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller,;marshal::v6::JournalTopicAckMarshaller,  
 545 2161

activemq::wireformat::openwire::marshal::v6::ActiveMQTraceMarshaller, 2184  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTransactionIdMarshaller, 2203  
 activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormatInfoMarshaller, 2230  
 activemq::wireformat::openwire::marshal::v6::BaseDataStreamMarshaller, 2264  
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller, 2312  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller, 2528  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2588  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2596  
 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2645  
 activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2675  
 activemq::wireformat::openwire::marshal::v6::MessageObjectMarshaller, 2721  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2755  
 activemq::wireformat::openwire::marshal::v6::PrivateQueueInfoMarshaller, 2871  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller, 3005  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3036  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3069  
 activemq::wireformat::openwire::marshal::v6::RemoteInfoMarshaller, 3147  
 activemq::wireformat::openwire::marshal::v6::RemoteSubscriptionInfoMarshaller, 3183  
 activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller, 3214  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3261  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3334  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3353  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3417  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3638  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3782

584	2223
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller,	activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller,
617	2251
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller,	activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller,
646	2285
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,	activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller,
674	2332
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller,
746	2544
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller,
842	2584
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller,
873	2613
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller,
1252	2650
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::MessageMarshaller,
1284	2672
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller,
1315	2718
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller,
1345	2771
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller,
1388	2894
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller,
1416	3010
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller,
1449	3041
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller,
1477	3058
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller,
1510	3155
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller,
1575	3171
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller,
1710	3203
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::ResponseMarshaller,
1743	3257
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller,
1827	3346
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller,
1921	3362
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller,
2075	3426
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller,
2141	3626
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller,
2170	3768
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller,
2192	3795
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller,

3941	1464
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller
3978	1498
activemq::wireformat::openwire::marshal::v2::ActiveMQBinaryMessageMarshaller	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller
192	1563
activemq::wireformat::openwire::marshal::v2::ActiveMQBinaryMessageMarshaller	activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller
242	1698
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v2::DiscoveryEventManager
322	1731
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller
362	1811
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller
389	1909
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller
435	2063
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller
478	2125
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller
541	2154
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller
568	2176
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller
596	2207
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller
625	2235
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
658	2273
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v2::LocalTransactionMarshaller
686	2316
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
766	2532
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
854	2568
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
885	2601
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller
1264	2630
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v2::MessageMarshaller
1272	2663
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller
1303	2702
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFileMarshaller
1332	2751
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller
1375	2876
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller
1404	2990
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller
1436	3021
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller

3054	666
activemq::wireformat::openwire::marshal::v2::ActiveMQInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller,
3143	732
activemq::wireformat::openwire::marshal::v2::ActiveMQSubscriptionInfoMarshaller,	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,
3180	834
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller,
3207	864
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller,
3243	1244
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller,
3326	1276
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller,
3370	1307
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller,
3422	1337
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller,
3642	1380
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,
3772	1408
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,
3811	1441
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller,
3933	1469
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller,
3970	1502
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller,
179	1567
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller,
222	1702
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller,
306	1735
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller,
346	1815
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller,
373	1913
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller,
418	2067
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller,
462	2133
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller,
525	2158
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,
552	2180
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller,
580	2211
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller,
609	2239
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller,
637	2269
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller,

2320	230
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
2536	314
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
2572	354
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
2605	381
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller
2642	427
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
2659	470
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMarshaller
2710	533
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
2763	560
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
2885	588
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
2998	613
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
3029	642
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
3066	670
activemq::wireformat::openwire::marshal::v3::ActiveMQWireMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQWireMarshaller
3151	739
activemq::wireformat::openwire::marshal::v3::ActiveMQSubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQSubscriptionInfoMarshaller
3176	838
activemq::wireformat::openwire::marshal::v3::ActiveMQBrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBrokerInfoMarshaller
3211	869
activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQConnectionControlMarshaller
3253	1248
activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQConnectionErrorMarshaller
3342	1280
activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQConnectionIdMarshaller
3366	1311
activemq::wireformat::openwire::marshal::v3::ActiveMQConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQConnectionInfoMarshaller
3434	1341
activemq::wireformat::openwire::marshal::v3::ActiveMQConsumerControlMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQConsumerControlMarshaller
3622	1384
activemq::wireformat::openwire::marshal::v3::ActiveMQConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQConsumerIdMarshaller
3775	1412
activemq::wireformat::openwire::marshal::v3::ActiveMQConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQConsumerInfoMarshaller
3799	1445
activemq::wireformat::openwire::marshal::v3::ActiveMQControlCommandMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQControlCommandMarshaller
3945	1473
activemq::wireformat::openwire::marshal::v3::ActiveMQDataArrayResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDataArrayResponseMarshaller
3982	1506
activemq::wireformat::openwire::marshal::v4::ActiveMQDataResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDataResponseMarshaller
188	1571
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationInfoMarshaller

1706	3199
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller
1739	3239
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller
1823	3330
activemq::wireformat::openwire::marshal::v4::ClusterWireFormatMarshaller	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller
1917	3374
activemq::wireformat::openwire::marshal::v4::IllegalResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller
2071	3438
activemq::wireformat::openwire::marshal::v4::InvalidWireFormatMarshaller	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller
2137	3634
activemq::wireformat::openwire::marshal::v4::InvalidTopicNameMarshaller	activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller
2166	3779
activemq::wireformat::openwire::marshal::v4::InvalidTraceIdMarshaller	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller
2188	3807
activemq::wireformat::openwire::marshal::v4::InvalidTransactionalMarshaller	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller
2219	3937
activemq::wireformat::openwire::marshal::v4::KeepAliveMarshaller	activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller
2243	3974
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
2281	196
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
2328	234
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller
2540	318
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
2580	358
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
2609	385
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
2634	431
activemq::wireformat::openwire::marshal::v4::MessageMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
2667	474
activemq::wireformat::openwire::marshal::v4::MessageRuleMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
2714	537
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller
2767	564
activemq::wireformat::openwire::marshal::v4::PersistentWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
2889	592
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
2994	621
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
3025	650
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
3049	678
activemq::wireformat::openwire::marshal::v4::RemoveWireMarshaller	activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller
3163	752
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller
3192	846
activemq::wireformat::openwire::marshal::v4::ReplyWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller

877	2618
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	2638
1256	2638
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	2654
1288	2654
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	2706
1319	2706
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	2759
1349	2759
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	2881
1392	2881
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3002
1420	3002
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3033
1453	3033
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3062
1481	3062
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3159
1514	3159
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3188
1555	3188
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3219
1718	3219
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3248
1747	3248
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3338
1819	3338
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3358
1925	3358
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3430
2079	3430
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3630
2129	3630
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3764
2150	3764
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3791
2196	3791
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3925
2215	3925
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	3986
2247	3986
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	200
2277	200
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	238
2324	238
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	326
2548	326
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	366
2576	366
activemq::wireformat::openwire::marshal::v5::ActiveMQConnectionFactoryMarshaller	



393	1905
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v6::IntegerResponseMarshaller
439	2059
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueFormatOpenWire::marshal::v6::JournalQueueAckMarshaller	
482	2121
activemq::wireformat::openwire::marshal::v6::ActiveMQSimpleMessageMarshaller	activemq::wireformat::openwire::marshal::v6::JournalTopicAckMarshaller
545	2162
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v6::JournalTraceMarshaller
572	2184
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v6::JournalTransactionMarshaller
600	2203
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v6::KeepAliveInfoMarshaller
629	2230
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v6::LastPartialCommandMarshaller
654	2264
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller
682	2312
activemq::wireformat::openwire::marshal::v6::ActiveMQWireIdMarshaller	activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller
759	2528
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::MessageDispatchMarshaller	
850	2588
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::MessageDispatchNotificationMarshaller	
881	2597
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::MessageIdMarshaller	
1260	2646
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::MessageMarshaller	
1292	2676
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::MessagePullMarshaller	
1323	2722
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::NetworkBridgeFilterMarshaller	
1353	2755
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::PartialCommandMarshaller	
1396	2872
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::ProducerAckMarshaller	
1424	3006
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::ProducerIdMarshaller	
1457	3037
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::ProducerInfoMarshaller	
1485	3070
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::RemoveInfoMarshaller	
1518	3147
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller	
1559	3184
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::ReplayCommandMarshaller	
1714	3215
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::ResponseMarshaller	
1727	3262
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::SessionIdMarshaller	
1806	3334
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::SessionInfoMarshaller	

3354 mapValue  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 2958  
 3418 mark  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 2958  
 3638 decaf::io::BufferedInputStream, 897  
 activemq::wireformat::openwire::marshal::v6::TransactionByteMarshaller, 989  
 3783 decaf::io::FilterInputStream, 1858  
 activemq::wireformat::openwire::marshal::v6::TransactionInputMarshaller, 2006  
 3803 decaf::io::PushbackInputStream, 3090  
 activemq::wireformat::openwire::marshal::v6::WireFormatReaderMarshaller, 3111  
 3929 decaf::nio::Buffer, 891  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 3966 decaf::util::zip::InflaterInputStream, 1999  
 looseUnmarshalBrokerError MarkBlock  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 779 MarkBlockLogger  
 looseUnmarshalByteArray decaf::util::logging::MarkBlockLogger, 2490  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 779 markSupported  
 looseUnmarshalCachedObject decaf::io::BufferedInputStream, 897  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 780 decaf::io::FilterInputStream, 1858  
 looseUnmarshalConstByteArray decaf::io::InputStream, 2006  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 780 decaf::io::Reader, 3111  
 looseUnmarshalLong decaf::util::zip::InflaterInputStream, 2000  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 780 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2953  
 looseUnmarshalNestedObject 2953  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 781 decaf::StreamMarshaller, 2844  
 activemq::wireformat::openwire::OpenWireFormat, 2844  
 2844 activemq::wireformat::openwire::utils::BooleanStream, 819  
 looseUnmarshalString activemq::wireformat::stomp::StompWireFormat, 859  
 activemq::wireformat::openwire::marshal::v6::WireFormatWriterMarshaller, 3111  
 781 activemq::wireformat::WireFormat, 3909  
 lowestOneBit marshalledProperties  
 decaf::lang::Integer, 2045 activemq::commands::Message, 2491  
 decaf::lang::Long, 2383 marshalledSize  
 activemq::wireformat::openwire::utils::BooleanStream, 819  
 MalformedURLException decaf::net::MalformedURLException, 2417  
 2418 marshallingSupport  
 activemq::util::MarshallingSupport, 2452  
 manageable marshalList  
 activemq::commands::ConnectionInfo, 1330  
 1330 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2953  
 Map marshalMap  
 decaf::util::Map, 2420 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2953  
 MAP\_TYPE 2954  
 activemq::util::PrimitiveValueNode, 2964  
 marshalPrimitive

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

activemq::core::MessageDispatchChannel, 2561  
 MessageDispatchMarshaller  
   activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2583  
   activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2567  
   activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2571  
   MessageMarshaller  
   activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2579  
   activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2575  
   activemq::wireformat::openwire::marshal::v6::MessageDispatchMarshaller, 2587  
 MessageDispatchNotification  
   activemq::commands::MessageDispatchNotification, 2592  
 MessageDispatchNotificationMarshaller  
   activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2612  
   activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2600  
   activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2604  
   activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2608  
   activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2617  
   activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller, 2596  
 MessageEOFException  
   cms::MessageEOFException, 2622  
 MessageFormatException  
   cms::MessageFormatException, 2623  
 MessageId  
   activemq::commands::MessageId, 2625  
 messageId  
   activemq::commands::JournalTopicAck, 2147  
   activemq::commands::Message, 2492  
   activemq::commands::MessageDispatchNotification, 2595  
   activemq::commands::MessagePull, 2699  
 MessageIdMarshaller  
   activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2649  
   activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2629  
   activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2641  
   activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2633  
   activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2637  
   activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller, 2645  
   MessageMarshaller  
   activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2671  
   activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2662  
   activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2658  
   activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2667  
   activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2654  
   activemq::wireformat::openwire::marshal::v6::MessageMarshaller, 2675  
   MessageNotReadableException  
   cms::MessageNotReadableException, 2679  
   MessageNotWritableException  
   cms::MessageNotWritableException, 2681  
   MessagePropertyInterceptor  
   activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2691  
   MessagePull  
   activemq::commands::MessagePull, 2696  
 MessagePullMarshaller  
   activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2717  
   activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2701  
   activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2709  
   activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2713  
   activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2705  
   activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller, 2721  
   messageSequenceId  
   MessageTopicMarshaller, JournalTopicAck, 2147  
   MessageIdMarshaller, internal\_state, 2084

---

MethodName	inflate.h, 4424
activemq::commands::BrokerError::StackTraceElement,	
3521	gz_header_s, 1939
MICROSECONDS	name_max
decaf::util::concurrent::TimeUnit, 3757	gz_header_s, 1939
MILLISECONDS	NAME_STATE
decaf::util::concurrent::TimeUnit, 3757	deflate.h, 4420
min	nameUUIDFromBytes
decaf::lang::Math, 2462–2464	decaf::util::UUID, 3903, 3904
MIN_LOOKAHEAD	NaN
deflate.h, 4420	decaf::lang::Double, 1762
MIN_MATCH	decaf::lang::Float, 1876
zutil.h, 4439	NANOSECONDS
MIN_PRIORITY	decaf::util::concurrent::TimeUnit, 3757
decaf::lang::Thread, 3716	nanoTime
MIN_RADIX	decaf::lang::System, 3676
decaf::lang::Character, 1076	narrow
MIN_VALUE	activemq::transport::failover::FailoverTransport,
decaf::lang::Byte, 927	1841
decaf::lang::Character, 1076	activemq::transport::IOTransport, 2109
decaf::lang::Double, 1762	activemq::transport::mock::MockTransport,
decaf::lang::Float, 1876	2729
decaf::lang::Integer, 2054	activemq::transport::Transport, 3822
decaf::lang::Long, 2392	activemq::transport::TransportFilter, 3831
decaf::lang::Short, 3389	ncode
MINUTES	inflate_state, 1984
decaf::util::concurrent::TimeUnit, 3757	ndist
MockTransport	inflate_state, 1984
activemq::transport::mock::MockTransport,	needsDictionary
2726	decaf::util::zip::Inflater, 1990
mode	needsInput
gz_state, 1940	decaf::util::zip::Deflater, 1677
inflate_state, 1984	decaf::util::zip::Inflater, 1990
modifiedUtf8ToAscii	NEGATIVE_INFINITY
activemq::util::MarshallingSupport, 2452	decaf::lang::Double, 1762
msg	decaf::lang::Float, 1876
gz_state, 1940	Network
z_stream_s, 3991	decaf::internal::net::Network, 2745
Mutex	NetworkBridgeFilter
decaf::util::concurrent::Mutex, 2737	activemq::commands::NetworkBridgeFilter,
mutex	2747
decaf::util::AbstractCollection, 160	NetworkBridgeFilterMarshaller
decaf::util::concurrent::ConditionHandle,	activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller
1227	2770
decaf::util::concurrent::MutexHandle, 2741,	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller
2742	2750
MutexHandle	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller
decaf::util::concurrent::MutexHandle, 2741	2762
NAME	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller
	2766

---

activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2758  
 nextIndex  
 activemq::wireformat::openwire::marshal::v6::NetworkBridgeFilterMarshaller, 2754  
 nextInt  
 networkBrokerId  
 decaf::util::Random, 3104, 3105  
 activemq::commands::NetworkBridgeFilter, 2749  
 nextLong  
 decaf::util::Random, 3105  
 networkConnection  
 nextMessage  
 activemq::commands::BrokerInfo, 862  
 activemq::core::ActiveMQQueueBrowser, 459  
 networkConsumerPath  
 cms::MessageEnumeration, 2621  
 activemq::commands::ConsumerInfo, 1433  
 nextToken  
 decaf::util::StringTokenizer, 3614, 3615  
 networkProperties  
 nice\_match  
 activemq::commands::BrokerInfo, 862  
 networkSubscription  
 internal\_state, 2084  
 activemq::commands::ConsumerInfo, 1434  
 nlen  
 inflate\_state, 1984  
 networkTTL  
 NO\_COMPRESSION  
 activemq::commands::NetworkBridgeFilter, 2749  
 decaf::util::zip::Deflater, 1681  
 NO\_MAXIMUM\_REDELIVERIES  
 NEW  
 activemq::core::RedeliveryPolicy, 3126  
 decaf::lang::Thread, 3710  
 node  
 newCondition  
 decaf::util::UUID, 3904  
 decaf::util::concurrent::locks::Lock, 2339  
 noLocal  
 decaf::util::concurrent::locks::ReentrantLock, 3131  
 activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3121  
 newThread  
 activemq::commands::ConsumerInfo, 1434  
 decaf::util::concurrent::ThreadFactory, 3717  
 NON\_PERSISTENT  
 cms::DeliveryMode, 1688  
 next  
 activemq::transport::TransportFilter, 3835  
 NoRangeAcks  
 decaf::security::SecureRandom, 3272  
 activemq::commands::ConsumerInfo, 1434  
 decaf::util::Iterator, 2115  
 NORM\_PRIORITY  
 decaf::util::Random, 3102  
 decaf::lang::Thread, 3716  
 gz\_state, 1940  
 inflate\_state, 1984  
 normalize  
 next\_in  
 decaf::net::URI, 3861  
 z\_stream\_s, 3991  
 NoRouteToHostException  
 next\_out  
 decaf::net::NoRouteToHostException, 2774, 2775  
 z\_stream\_s, 3991  
 NoSuchAlgorithmException  
 nextBoolean  
 decaf::util::Random, 3103  
 decaf::security::NoSuchAlgorithmException, 2776, 2777  
 nextBytes  
 decaf::security::SecureRandom, 3273  
 NoSuchElementException  
 decaf::util::Random, 3103  
 decaf::lang::exceptions::NoSuchElementException, 2779, 2780  
 nextDouble  
 decaf::util::Random, 3103  
 NoSuchProviderException  
 nextFloat  
 decaf::security::NoSuchProviderException, 2782, 2783  
 decaf::util::Random, 3104  
 notify  
 nextGaussian

activemq::core::MessageDispatchChannel, 2563  
 decaf::internal::util::concurrent::ConditionImpl, 1229  
 decaf::internal::util::concurrent::SynchronizableImpl, 3656  
 decaf::io::InputStream, 2006  
 decaf::io::OutputStream, 2860  
 decaf::util::AbstractCollection, 154  
 decaf::util::concurrent::ConcurrentStlMap, 1211  
 decaf::util::concurrent::Mutex, 2737  
 decaf::util::concurrent::Synchronizable, 3646  
 decaf::util::StlMap, 3551  
 decaf::util::StlQueue, 3561  
 notifyAll  
 activemq::core::MessageDispatchChannel, 2563  
 decaf::internal::util::concurrent::ConditionImpl, 1229  
 decaf::internal::util::concurrent::SynchronizableImpl, 3656  
 decaf::io::InputStream, 2007  
 decaf::io::OutputStream, 2860  
 decaf::util::AbstractCollection, 155  
 decaf::util::concurrent::ConcurrentStlMap, 1212  
 decaf::util::concurrent::Mutex, 2738  
 decaf::util::concurrent::Synchronizable, 3647  
 decaf::util::StlMap, 3551  
 decaf::util::StlQueue, 3561  
 Null  
 decaf::util::logging, 143  
 NULL\_TYPE  
 activemq::util::PrimitiveValueNode, 2963  
 activemq::wireformat::openwire::OpenWireCommand, 2849  
 NullPointerException  
 decaf::lang::exceptions::NullPointerException, 2784, 2785  
 NUM\_OPTIONS  
 activemq::core::ActiveMQConstants, 281  
 NUM\_PARAMS  
 activemq::core::ActiveMQConstants, 281  
 NumberFormatException  
 decaf::lang::exceptions::NumberFormatException, 2789, 2790  
 numberOfLeadingZeros  
 decaf::lang::Integer, 2045  
 decaf::lang::Long, 2384  
 numberOfTrailingZeros  
 decaf::lang::Integer, 2045  
 decaf::lang::Long, 2384  
 numWaiting  
 decaf::util::concurrent::ConditionHandle, 1227  
 numWake  
 decaf::util::concurrent::ConditionHandle, 1227  
 objectId  
 activemq::commands::RemoveInfo, 3141  
 OF  
 deflate.h, 4421  
 gzguts.h, 4423  
 infast.h, 4423  
 infrees.h, 4426  
 zconf.h, 4429  
 zlib.h, 4435–4437  
 zutil.h, 4440  
 OFF  
 decaf::util::logging::Level, 2295  
 Off  
 decaf::util::logging, 143  
 offer  
 decaf::util::concurrent::BlockingQueue, 808  
 decaf::util::concurrent::SynchronousQueue, 3665, 3666  
 decaf::util::PriorityQueue, 2980  
 decaf::util::Queue, 3096  
 offset  
 decaf::internal::nio::CharArrayBuffer, 1089  
 inflate\_state, 1984  
 WireFormat  
 activemq::core::ActiveMQConnection, 257  
 activemq::transport::correlator::ResponseCorrelator, 3234  
 activemq::transport::DefaultTransportListener, 1671  
 activemq::transport::failover::FailoverTransportListener, 1849  
 activemq::transport::inactivity::InactivityMonitor,  
 activemq::transport::logging::LoggingTransport, 2361

activemq::transport::mock::InternalCommandListener, 2086  
 activemq::transport::TransportFilter, 3830  
 activemq::transport::TransportListener, 3836  
 activemq::wireformat::openwire::OpenWireFormatNegotiator, 2852  
 oneway  
 activemq::core::ActiveMQConnection, 257  
 activemq::core::ActiveMQSession, 499  
 activemq::transport::correlator::ResponseCorrelator, 3234  
 activemq::transport::failover::FailoverTransport, 1842  
 activemq::transport::inactivity::InactivityMonitor, 1966  
 activemq::transport::IOTransport, 2109  
 activemq::transport::logging::LoggingTransport, 2362  
 activemq::transport::mock::MockTransport, 2729  
 activemq::transport::Transport, 3822  
 activemq::transport::TransportFilter, 3832  
 activemq::wireformat::openwire::OpenWireFormatNegotiator, 2852  
 onException  
 activemq::core::ActiveMQConnection, 258  
 activemq::transport::DefaultTransportListener, 1671  
 activemq::transport::failover::BackupTransport, 719  
 activemq::transport::failover::FailoverTransportListener, 1850  
 activemq::transport::inactivity::InactivityMonitor, 1966  
 activemq::transport::TransportFilter, 3832  
 activemq::transport::TransportListener, 3837  
 cms::ExceptionListener, 1801  
 onMessage  
 cms::MessageListener, 2652  
 onProducerAck  
 activemq::core::ActiveMQProducer, 445  
 onPropertiesReset  
 decaf::util::logging::PropertiesChangeListener, 3083  
 onPropertyChanged  
 decaf::util::logging::PropertiesChangeListener, 3083  
 activemq::state::Tracked, 3759  
 onSend  
 activemq::commands::ActiveMQBytesMessage, 207  
 activemq::commands::ActiveMQMessageTemplate, 407  
 activemq::commands::ActiveMQStreamMessage, 510  
 activemq::commands::Message, 2487  
 onTaskComplete  
 decaf::util::concurrent::TaskListener, 3679  
 onTaskCompleted  
 decaf::util::concurrent::PooledThreadListener, 2921  
 decaf::util::concurrent::ThreadPool, 3722  
 onTaskException  
 decaf::util::concurrent::PooledThreadListener, 2921  
 decaf::util::concurrent::TaskListener, 3680  
 decaf::util::concurrent::ThreadPool, 3722  
 onTaskStarted  
 decaf::util::concurrent::PooledThreadListener, 2921  
 decaf::util::concurrent::ThreadPool, 3722  
 onTransportException  
 activemq::transport::correlator::ResponseCorrelator, 3235  
 activemq::wireformat::openwire::OpenWireFormatNegotiator, 2853  
 opaque  
 code, 1154  
 z\_stream\_s, 3991  
 OPEN\_FAILURE  
 decaf::util::logging::ErrorManager, 1793  
 OpenSSLContextSpi  
 decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2793  
 OpenSSLServerSocket  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2799  
 OpenSSLServerSocketFactory  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocketFactory, 2805  
 OpenSSLSocket  
 decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2795



- decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2812, 2813
- OpenSSLSocketException
  - decaf::internal::net::ssl::openssl::OpenSSLSocketException, 2822–2824
- OpenSSLSocketFactory
  - decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 2795
  - decaf::internal::net::ssl::openssl::OpenSSLSocketFactory, 2828
- OpenSSLSocketInputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocketInputStream, 2833
- OpenSSLSocketOutputStream
  - decaf::internal::net::ssl::openssl::OpenSSLSocketOutputStream, 2836
- OpenWireFormat
  - activemq::wireformat::openwire::OpenWireFormat, 2839
- OpenWireFormatFactory
  - activemq::wireformat::openwire::OpenWireFormatFactory, 2850
- OpenWireFormatNegotiator
  - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2851
- OpenWireResponseBuilder
  - activemq::wireformat::openwire::OpenWireResponseBuilder, 2855
- operationType
  - activemq::commands::DestinationInfo, 1696
- operator<
  - activemq::commands::BrokerId, 831
  - activemq::commands::ConnectionId, 1300
  - activemq::commands::ConsumerId, 1401
  - activemq::commands::LocalTransactionId, 2309
  - activemq::commands::MessageId, 2627
  - activemq::commands::ProducerId, 3017
  - activemq::commands::SessionId, 3323
  - activemq::commands::TransactionId, 3762
  - activemq::commands::XATransactionId, 3963
  - decaf::lang::Boolean, 813
  - decaf::lang::Byte, 923
  - decaf::lang::Character, 1074, 1075
  - decaf::lang::Comparable, 1188
  - decaf::lang::Double, 1758
  - decaf::lang::Float, 1872
  - decaf::lang::Integer, 2046
  - decaf::lang::Long, 2384, 2385
  - decaf::lang::Short, 3385
  - decaf::net::URI, 3861
  - decaf::nio::ByteBuffer, 1012
  - decaf::nio::CharBuffer, 1100
  - decaf::nio::DoubleBuffer, 1781
  - decaf::nio::FloatBuffer, 1895
  - decaf::nio::IntBuffer, 2034
  - decaf::nio::LongBuffer, 2411
  - decaf::nio::ShortBuffer, 3408
  - decaf::util::concurrent::TimeUnit, 3751
  - decaf::util::Date, 1636
  - decaf::util::logging::Level, 2293
  - decaf::util::UUID, 3904
  - operator\*
    - decaf::lang::Pointer, 2900, 2901
  - operator()
    - decaf::lang::ArrayPointerComparator, 705
  - decaf::lang::PointerComparator, 2904
  - decaf::util::Comparator, 1190
  - decaf::util::comparators::Less, 2288
  - std::less< decaf::lang::ArrayPointer< T > >, 2289
  - std::less< decaf::lang::Pointer< T > >, 2290
  - operator->
    - decaf::lang::Pointer, 2901
  - operator=
    - activemq::cmsutil::CachedConsumer, 1043
    - activemq::cmsutil::CachedProducer, 1047
    - activemq::cmsutil::CmsAccessor, 1127
    - activemq::cmsutil::CmsDestinationAccessor, 1129
    - activemq::cmsutil::CmsTemplate, 1147
    - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3014
    - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3121
    - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3222
    - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3223
    - activemq::cmsutil::CmsTemplate::SendExecutor, 3291
    - activemq::cmsutil::DynamicDestinationResolver, 1787
    - activemq::cmsutil::PooledSession, 2917

activemq::cmsutil::ResourceLifecycleManager, 3227  
 activemq::cmsutil::SessionPool, 3377  
 activemq::commands::BrokerId, 831  
 activemq::commands::ConnectionId, 1300  
 activemq::commands::ConsumerId, 1401  
 activemq::commands::LocalTransactionId, 2309  
 activemq::commands::MessageId, 2627  
 activemq::commands::ProducerId, 3018  
 activemq::commands::SessionId, 3323  
 activemq::commands::TransactionId, 3762  
 activemq::commands::XATransactionId, 3963  
 activemq::library::ActiveMQCPP, 293  
 activemq::util::PrimitiveValueNode, 2971  
 decaf::lang::ArrayPointer, 702  
 decaf::lang::Exception, 1799  
 decaf::lang::Pointer, 2901, 2902  
 decaf::util::AbstractCollection, 155  
 decaf::util::Date, 1636  
 decaf::util::logging::LogManager, 2368  
 decaf::util::PriorityQueue, 2981  
 decaf::util::Properties, 3079  
 operator==  
 activemq::commands::BrokerId, 831  
 activemq::commands::ConnectionId, 1300  
 activemq::commands::ConsumerId, 1401  
 activemq::commands::LocalTransactionId, 2309  
 activemq::commands::MessageId, 2627  
 activemq::commands::ProducerId, 3018  
 activemq::commands::SessionId, 3323  
 activemq::commands::TransactionId, 3762  
 activemq::commands::XATransactionId, 3963  
 activemq::util::PrimitiveValueNode, 2971  
 decaf::lang, 133, 134  
 decaf::lang::ArrayPointer, 702, 704  
 decaf::lang::Boolean, 814  
 decaf::lang::Byte, 924  
 decaf::lang::Character, 1075  
 decaf::lang::Comparable, 1188  
 decaf::lang::Double, 1758, 1759  
 decaf::lang::Float, 1872, 1873  
 decaf::lang::Integer, 2046, 2047  
 decaf::lang::Long, 2385  
 decaf::lang::Pointer, 2902, 2903  
 decaf::lang::Short, 3386  
 decaf::net::URI, 3862  
 decaf::nio::ByteBuffer, 1012  
 decaf::nio::CharBuffer, 1100  
 decaf::nio::DoubleBuffer, 1781  
 decaf::nio::FloatBuffer, 1895  
 decaf::nio::IntBuffer, 2034  
 decaf::nio::LongBuffer, 2411  
 decaf::nio::ShortBuffer, 3408  
 decaf::util::concurrent::TimeUnit, 3752  
 decaf::util::Date, 1636  
 decaf::util::logging::Level, 2293  
 decaf::util::UUID, 3905  
 operator[]  
 activemq::wireformat::openwire::utils::HexTable, 1947, 1948  
 decaf::internal::util::ByteArrayAdapter, 943, 944  
 decaf::lang::ArrayPointer, 702  
 opt\_len  
 internal\_state, 2084  
 optimizedAcknowledge  
 activemq::commands::ConsumerInfo, 1434  
 options  
 activemq::commands::ActiveMQDestination, 303  
 ordered  
 activemq::commands::ActiveMQDestination, 303  
 orderedTarget  
 activemq::commands::ActiveMQDestination, 303  
 originalDestination  
 activemq::commands::Message, 2492  
 originalTransactionId  
 activemq::commands::Message, 2492  
 OS  
 inflate.h, 4424  
 os  
 gz\_header\_s, 1939  
 OS\_CODE  
 zutil.h, 4439  
 out  
 decaf::io::FileDescriptor, 1852  
 gz\_state, 1940  
 OutputStream  
 decaf::io::OutputStream, 2858  
 outputStream  
 decaf::io::FilterOutputStream, 1864  
 OutputStreamWriter  
 decaf::io::OutputStreamWriter, 2865

- own
  - decaf::io::FilterInputStream, 1860
  - decaf::io::FilterOutputStream, 1864
- ownDeflater
  - decaf::util::zip::DeflaterOutputStream, 1686
- ownInflater
  - decaf::util::zip::InflaterInputStream, 2002
- PARAM\_CLIENTID
  - activemq::core::ActiveMQConstants, 281
- PARAM\_PASSWORD
  - activemq::core::ActiveMQConstants, 281
- PARAM\_USERNAME
  - activemq::core::ActiveMQConstants, 281
- parent
  - activemq::cmsutil::CmsTemplate::ProducerExecutor, 3014
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3121
- park
  - decaf::util::concurrent::locks::LockSupport, 2343
- parkNanos
  - decaf::util::concurrent::locks::LockSupport, 2343
- parkUntil
  - decaf::util::concurrent::locks::LockSupport, 2344
- parse
  - decaf::internal::util::HexStringParser, 1946
  - decaf::util::logging::Level, 2293
- parseAuthority
  - decaf::internal::net::URIHelper, 3871
- parseBoolean
  - decaf::lang::Boolean, 814
- parseByte
  - decaf::lang::Byte, 924, 925
- parseComposite
  - activemq::util::URISupport, 3878
- parseDouble
  - decaf::internal::util::HexStringParser, 1946
  - decaf::lang::Double, 1759
- parseFloat
  - decaf::internal::util::HexStringParser, 1946
  - decaf::lang::Float, 1873
- parseInt
  - decaf::lang::Integer, 2047, 2048
- parseLong
  - decaf::lang::Long, 2386
- parseQuery
  - activemq::util::URISupport, 3878, 3879
- parseServerAuthority
  - decaf::net::URI, 3862
- parseShort
  - decaf::lang::Short, 3386, 3387
- parseURI
  - decaf::internal::net::URIHelper, 3872
- parseURL
  - activemq::util::URISupport, 3879
- PartialCommand
  - activemq::commands::PartialCommand, 2867
- PartialCommandMarshaller
  - activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2892
  - activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2875
  - activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2884
  - activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2888
  - activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2879
  - activemq::wireformat::openwire::marshal::v6::PartialCommandMarshaller, 2871
- password
  - activemq::commands::ConnectionInfo, 1330
- path
  - gz\_state, 1940
- peek
  - activemq::core::MessageDispatchChannel, 2563
  - decaf::internal::util::TimerTaskHeap, 3747
  - decaf::util::concurrent::SynchronousQueue, 3667
  - decaf::util::PriorityQueue, 2981
  - decaf::util::Queue, 3097
- peerBrokerInfos
  - activemq::commands::BrokerInfo, 862
- pending
  - internal\_state, 2084
- pending\_buf
  - internal\_state, 2084
- pending\_buf\_size
  - internal\_state, 2084
- pending\_out
  - internal\_state, 2084
- PERSISTENT

- cms::DeliveryMode, 1688
- persistent
  - activemq::commands::Message, 2492
- physicalName
  - activemq::commands::ActiveMQDestination, 4439
- PI
  - decaf::lang::Math, 2472
- Pointer
  - decaf::lang::Pointer, 2898, 2899
- PointerType
  - decaf::lang::ArrayPointer, 699
  - decaf::lang::Pointer, 2898
- poll
  - decaf::util::concurrent::BlockingQueue, 809
  - decaf::util::concurrent::SynchronousQueue, 3667
  - decaf::util::PriorityQueue, 2982
  - decaf::util::Queue, 3097
- PooledSession
  - activemq::cmsutil::PooledSession, 2907
- PooledThread
  - decaf::util::concurrent::PooledThread, 2919
- pop
  - decaf::util::StlQueue, 3561
- port
  - decaf::net::SocketImpl, 3481
- PortUnreachableException
  - decaf::net::PortUnreachableException, 2923, 2924
- Pos
  - deflate.h, 4421
- pos
  - gz\_state, 1940
- Posf
  - deflate.h, 4421
- position
  - decaf::nio::Buffer, 891, 892
- POSITIVE\_INFINITY
  - decaf::lang::Double, 1762
  - decaf::lang::Float, 1876
- pow
  - decaf::lang::Math, 2464
- prefetch
  - activemq::commands::ConsumerControl, 1373
- PrefetchPolicy
  - activemq::core::PrefetchPolicy, 2926
- prefetchSize
  - activemq::commands::ConsumerInfo, 1434
- PRESET\_DICT
- prev
  - internal\_state, 2084
- prev\_length
  - internal\_state, 2084
- prev\_match
  - internal\_state, 2084
- previous
  - decaf::util::ListIterator, 2305
- previousIndex
  - decaf::util::ListIterator, 2306
- PrimitiveList
- PrimitiveMap
  - activemq::util::PrimitiveMap, 2943
- PrimitiveType
  - activemq::util::PrimitiveValueNode, 2963
- PrimitiveTypesMarshaller
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2952
- PrimitiveValueConverter
  - activemq::util::PrimitiveValueConverter, 2959
- PrimitiveValueNode
  - activemq::util::PrimitiveValueNode, 2964–2966
- printStackTrace
  - cms::CMSEException, 1133
  - decaf::lang::Exception, 1799
  - decaf::lang::Throwable, 3727
- priority
  - activemq::commands::ConsumerInfo, 1434
  - activemq::commands::Message, 2492
- PriorityQueue
  - decaf::util::PriorityQueue, 2978, 2979
- PriorityQueueIterator
  - decaf::util::PriorityQueue, 2984
- processBeginTransaction
  - activemq::state::CommandVisitor, 1173
  - activemq::state::CommandVisitorAdapter, 1182
  - activemq::state::ConnectionStateTracker, 1363
- processBrokerError
  - activemq::state::CommandVisitor, 1173

activemq::state::CommandVisitorAdapter, 1182	activemq::state::ConnectionStateTracker, 1364
processBrokerInfo	processEndTransaction
activemq::state::CommandVisitor, 1174	activemq::state::CommandVisitor, 1175
activemq::state::CommandVisitorAdapter, 1182	activemq::state::CommandVisitorAdapter, 1183
processCommitTransactionOnePhase	activemq::state::ConnectionStateTracker, 1364
activemq::state::CommandVisitor, 1174	processFlushCommand
activemq::state::CommandVisitorAdapter, 1182	activemq::state::CommandVisitor, 1175
activemq::state::ConnectionStateTracker, 1363	activemq::state::CommandVisitorAdapter, 1183
processCommitTransactionTwoPhase	processForgetTransaction
activemq::state::CommandVisitor, 1174	activemq::state::CommandVisitor, 1175
activemq::state::CommandVisitorAdapter, 1182	activemq::state::CommandVisitorAdapter, 1183
activemq::state::ConnectionStateTracker, 1363	processKeepAliveInfo
processConnectionControl	activemq::state::CommandVisitor, 1175
activemq::state::CommandVisitor, 1174	activemq::state::CommandVisitorAdapter, 1183
activemq::state::CommandVisitorAdapter, 1182	processMessage
processConnectionError	activemq::state::CommandVisitor, 1175
activemq::state::CommandVisitor, 1174	activemq::state::CommandVisitorAdapter, 1183
activemq::state::CommandVisitorAdapter, 1182	activemq::state::ConnectionStateTracker, 1364
processConnectionInfo	processMessageAck
activemq::state::CommandVisitor, 1174	activemq::state::CommandVisitor, 1175
activemq::state::CommandVisitorAdapter, 1182	activemq::state::CommandVisitorAdapter, 1183
activemq::state::ConnectionStateTracker, 1364	activemq::state::ConnectionStateTracker, 1364
processConsumerControl	processMessageDispatch
activemq::state::CommandVisitor, 1174	activemq::state::CommandVisitor, 1176
activemq::state::CommandVisitorAdapter, 1182	activemq::state::CommandVisitorAdapter, 1184
processConsumerInfo	processMessageDispatchNotification
activemq::state::CommandVisitor, 1174	activemq::state::CommandVisitor, 1176
activemq::state::CommandVisitorAdapter, 1183	activemq::state::CommandVisitorAdapter, 1184
activemq::state::ConnectionStateTracker, 1364	processMessagePull
processControlCommand	activemq::state::CommandVisitor, 1176
activemq::state::CommandVisitor, 1175	activemq::state::CommandVisitorAdapter, 1184
activemq::state::CommandVisitorAdapter, 1183	processPrepareTransaction
processDestinationInfo	activemq::state::CommandVisitor, 1176
activemq::state::CommandVisitor, 1175	activemq::state::CommandVisitorAdapter, 1184
activemq::state::CommandVisitorAdapter, 1183	activemq::state::ConnectionStateTracker, 1365

processProducerAck	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1176	1185
activemq::state::CommandVisitorAdapter,	processReplayCommand
1184	activemq::state::CommandVisitor, 1178
processProducerInfo	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1176	1185
activemq::state::CommandVisitorAdapter,	processResponse
1184	activemq::state::CommandVisitor, 1178
activemq::state::ConnectionStateTracker,	activemq::state::CommandVisitorAdapter,
1365	1185
processRecoverTransactions	processRollbackTransaction
activemq::state::CommandVisitor, 1176	activemq::state::CommandVisitor, 1178
activemq::state::CommandVisitorAdapter,	activemq::state::CommandVisitorAdapter,
1184	1185
processRemoveConnection	activemq::state::ConnectionStateTracker,
activemq::state::CommandVisitor, 1176	1366
activemq::state::CommandVisitorAdapter,	processSessionInfo
1184	activemq::state::CommandVisitor, 1178
activemq::state::ConnectionStateTracker,	activemq::state::CommandVisitorAdapter,
1365	1186
processRemoveConsumer	activemq::state::ConnectionStateTracker,
activemq::state::CommandVisitor, 1177	1366
activemq::state::CommandVisitorAdapter,	processShutdownInfo
1184	activemq::state::CommandVisitor, 1178
activemq::state::ConnectionStateTracker,	activemq::state::CommandVisitorAdapter,
1365	1186
processRemoveDestination	processTransactionInfo
activemq::state::CommandVisitor, 1177	activemq::state::CommandVisitor, 1178
activemq::state::CommandVisitorAdapter,	activemq::state::CommandVisitorAdapter,
1185	1186
activemq::state::ConnectionStateTracker,	processWireFormat
1365	activemq::state::CommandVisitor, 1178
processRemoveInfo	activemq::state::CommandVisitorAdapter,
activemq::state::CommandVisitor, 1177	1186
activemq::state::CommandVisitorAdapter,	PRODUCER_ADVISORY_PREFIX
1185	activemq::commands::ActiveMQDestination,
processRemoveProducer	303
activemq::state::CommandVisitor, 1177	ProducerAck
activemq::state::CommandVisitorAdapter,	activemq::commands::ProducerAck, 2985
1185	ProducerAckMarshaller
activemq::state::ConnectionStateTracker,	activemq::wireformat::openwire::marshal::v1::ProducerAckMar
1365	3009
processRemoveSession	activemq::wireformat::openwire::marshal::v2::ProducerAckMar
activemq::state::CommandVisitor, 1177	2989
activemq::state::CommandVisitorAdapter,	activemq::wireformat::openwire::marshal::v3::ProducerAckMar
1185	2997
activemq::state::ConnectionStateTracker,	activemq::wireformat::openwire::marshal::v4::ProducerAckMar
1366	2993
processRemoveSubscriptionInfo	activemq::wireformat::openwire::marshal::v5::ProducerAckMar
activemq::state::CommandVisitor, 1177	3001

activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3005  
 ProducerExecutor  
    activemq::cmsutil::CmsTemplate, 1153  
    activemq::cmsutil::CmsTemplate::ProducerExecutor, 3013  
 ProducerId  
    activemq::commands::ProducerId, 3016  
 producerId  
    activemq::commands::Message, 2492  
    activemq::commands::MessageId, 2628  
    activemq::commands::ProducerAck, 2987  
    activemq::commands::ProducerInfo, 3047  
 ProducerIdMarshaller  
    activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 3040  
    activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 3020  
    activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 3028  
    activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 3024  
    activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 3032  
    activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller, 3036  
 ProducerInfo  
    activemq::commands::ProducerInfo, 3044  
 ProducerInfoMarshaller  
    activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 3057  
    activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 3053  
    activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 3065  
    activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 3048  
    activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 3061  
    activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller, 3069  
 producerSequenceId  
    activemq::commands::MessageId, 2628  
 ProducerState  
    activemq::state::ProducerState, 3072  
 Properties  
    decaf::util::Properties, 3074  
 propertyExists  
    activemq::commands::ActiveMQMessageTemplate, 407  
    decaf::util::Properties, 3079  
    ProtocolException, 3084, 3085  
    providerGenerateSeed, 3276  
    decaf::internal::security::SecureRandomImpl, 3279  
    decaf::security::SecureRandomSpi, 3493  
    providerGetDefaultSSLParameters, 3493  
    decaf::net::ssl::SSLContextSpi, 3493  
    providerGetServerSocketFactory, 3494  
    decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 3494  
    decaf::net::ssl::SSLContextSpi, 3493  
    providerGetSocketFactory, 3494  
    decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 3494  
    decaf::net::ssl::SSLContextSpi, 3493  
    providerGetSupportedSSLParameters, 3494  
    decaf::net::ssl::SSLContextSpi, 3494  
    providerInit, 3494  
    decaf::internal::net::ssl::openssl::OpenSSLContextSpi, 3494  
    decaf::net::ssl::SSLContextSpi, 3494  
    providerNextBytes, 3276, 3277  
    decaf::internal::security::SecureRandomImpl, 3279  
    decaf::security::SecureRandomSpi, 3279  
    providerSetSeed, 3277  
    decaf::internal::security::SecureRandomImpl, 3277  
    decaf::security::SecureRandomSpi, 3279  
    publish, 3738  
    decaf::util::logging::ConsoleHandler, 1368  
    decaf::util::logging::Handler, 1943  
    decaf::util::logging::StreamHandler, 3594  
    purge, 3738  
    decaf::util::Timer, 3738  
    push, 3561  
    decaf::util::StlQueue, 3561  
    PushbackInputStream, 3088, 3089  
    decaf::io::PushbackInputStream, 3088, 3089  
    put, 977  
    decaf::internal::nio::ByteBuffer, 976, 1086, 1087

- decaf::internal::nio::DoubleArrayBuffer, 1771, 1772
- decaf::internal::nio::FloatArrayBuffer, 1885, 1886
- decaf::internal::nio::IntArrayBuffer, 2024, 2025
- decaf::internal::nio::LongArrayBuffer, 2401, 2402
- decaf::internal::nio::ShortArrayBuffer, 3399
- decaf::internal::util::ByteArrayAdapter, 944
- decaf::nio::ByteBuffer, 1012–1015
- decaf::nio::CharBuffer, 1100–1104
- decaf::nio::DoubleBuffer, 1781–1783
- decaf::nio::FloatBuffer, 1895–1897
- decaf::nio::IntBuffer, 2034–2036
- decaf::nio::LongBuffer, 2411–2413
- decaf::nio::ShortBuffer, 3408–3410
- decaf::util::concurrent::BlockingQueue, putIntAt 809
- decaf::util::concurrent::ConcurrentStlMap, 1212
- decaf::util::concurrent::SynchronousQueue, 3667
- decaf::util::Map, 2427
- decaf::util::StlMap, 3552
- put\_byte
  - deflate.h, 4420
- putAll
  - decaf::util::concurrent::ConcurrentStlMap, 1213
  - decaf::util::Map, 2428
  - decaf::util::StlMap, 3552
- putChar
  - decaf::internal::nio::ByteBuffer, 977, 978
  - decaf::internal::util::ByteArrayAdapter, 944
  - decaf::nio::ByteBuffer, 1015, 1016
- putDouble
  - decaf::internal::nio::ByteBuffer, 979
  - decaf::internal::util::ByteArrayAdapter, 945
  - decaf::nio::ByteBuffer, 1016, 1017
- putDoubleAt
  - decaf::internal::util::ByteArrayAdapter, 945
- putFloat
  - decaf::internal::nio::ByteBuffer, 979, 980
  - decaf::internal::util::ByteArrayAdapter, 946
  - decaf::nio::ByteBuffer, 1017, 1018
  - putFloatAt
  - decaf::internal::util::ByteArrayAdapter, 946
  - putIfAbsent
  - decaf::util::concurrent::ConcurrentMap, 1199
  - decaf::util::concurrent::ConcurrentStlMap, 1213
  - putInt
  - decaf::internal::nio::ByteBuffer, 980, 981
  - decaf::internal::util::ByteArrayAdapter, 946
  - decaf::nio::ByteBuffer, 1018, 1019
  - decaf::internal::util::ByteArrayAdapter, 947
  - putLong
  - decaf::internal::nio::ByteBuffer, 981, 982
  - decaf::internal::util::ByteArrayAdapter, 947
  - decaf::nio::ByteBuffer, 1019, 1020
  - putLongAt
  - decaf::internal::util::ByteArrayAdapter, 948
  - putShort
  - decaf::internal::nio::ByteBuffer, 982, 983
  - decaf::internal::util::ByteArrayAdapter, 948
  - decaf::nio::ByteBuffer, 1020, 1021
  - putShortAt
  - decaf::internal::util::ByteArrayAdapter, 949
- QUEUE
  - cms::Destination, 1689
- QUEUE\_PREFIX
  - activemq::wireformat::stomp::StompCommandConstants, 3576
- QUEUE\_QUALIFIED\_PREFIX
  - activemq::commands::ActiveMQDestination, 303
- queueTask



- decaf::util::concurrent::ThreadPool, 3722
- quotelllegal
- decaf::internal::net::URLEncoderDecoder, 3866
- Random
  - decaf::util::Random, 3102
- random
  - decaf::lang::Math, 2465
- randomUUID
  - decaf::util::UUID, 3905
- raw
  - gz\_state, 1940
- reached
  - decaf::net::InetAddress, 1982
- read
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2817
  - decaf::internal::net::tcp::TcpSocket, 3689
  - decaf::internal::util::ByteArrayAdapter, 949
  - decaf::io::InputStream, 2007, 2008
  - decaf::io::Reader, 3111–3113
  - decaf::lang::Readable, 3107
  - decaf::nio::CharBuffer, 1104
- readAsciiString
  - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 782
- readBoolean
  - activemq::commands::ActiveMQBytesMessage, 207
  - activemq::commands::ActiveMQStreamMessage, 511
  - activemq::wireformat::openwire::utils::BooleanStream, 820
  - cms::BytesMessage, 1027
  - cms::StreamMessage, 3597
  - decaf::io::DataInput, 1524
  - decaf::io::DataInputStream, 1534
- readByte
  - activemq::commands::ActiveMQBytesMessage, 207
  - activemq::commands::ActiveMQStreamMessage, 511
  - cms::BytesMessage, 1028
  - cms::StreamMessage, 3598
  - decaf::io::DataInput, 1525
  - decaf::io::DataInputStream, 1534
- readBytes
  - activemq::commands::ActiveMQBytesMessage, 208, 209
  - activemq::commands::ActiveMQStreamMessage, 512
  - cms::BytesMessage, 1028, 1029
  - cms::StreamMessage, 3598, 3599
  - readChar
    - activemq::commands::ActiveMQBytesMessage, 209
    - activemq::commands::ActiveMQStreamMessage, 513
    - cms::BytesMessage, 1030
    - cms::StreamMessage, 3600
    - decaf::io::DataInput, 1525
    - decaf::io::DataInputStream, 1535
  - ReadChecker
    - activemq::transport::inactivity::InactivityMonitor, 1967
    - activemq::transport::inactivity::ReadChecker, 3108
  - readConfiguration
    - decaf::util::logging::LogManager, 2368, 2369
  - readDouble
    - activemq::commands::ActiveMQBytesMessage, 210
    - activemq::commands::ActiveMQStreamMessage, 514
    - cms::BytesMessage, 1030
    - cms::StreamMessage, 3601
    - decaf::io::DataInput, 1525
    - decaf::io::DataInputStream, 1535
  - Reader
    - decaf::io::Reader, 3110
  - readFloat
    - activemq::commands::ActiveMQBytesMessage, 210
    - activemq::commands::ActiveMQStreamMessage, 514
    - cms::BytesMessage, 1031
    - cms::StreamMessage, 3601
    - decaf::io::DataInput, 1526
    - decaf::io::DataInputStream, 1535
  - readFully
    - decaf::io::DataInput, 1526, 1527
    - decaf::io::DataInputStream, 1536
  - readInt
    - activemq::commands::ActiveMQBytesMessage, 211

activemq::commands::ActiveMQStreamMessage, 515  
 cms::BytesMessage, 1031  
 cms::StreamMessage, 3602  
 decaf::io::DataInput, 1528  
 decaf::io::DataInputStream, 1537  
 readLine  
   decaf::io::DataInput, 1528  
   decaf::io::DataInputStream, 1537  
 readLock  
   decaf::util::concurrent::locks::ReadWriteLock, 3119  
 readLong  
   activemq::commands::ActiveMQBytesMessage, 211  
   activemq::commands::ActiveMQStreamMessage, 515  
   cms::BytesMessage, 1032  
   cms::StreamMessage, 3602  
   decaf::io::DataInput, 1529  
   decaf::io::DataInputStream, 1538  
 readOnly  
   decaf::internal::nio::CharArrayBuffer, 1089  
 readonly  
   decaf::io::FileDescriptor, 1853  
 ReadOnlyBufferException  
   decaf::nio::ReadOnlyBufferException, 3115, 3116  
 readShort  
   activemq::commands::ActiveMQBytesMessage, 212  
   activemq::commands::ActiveMQStreamMessage, 516  
   cms::BytesMessage, 1032  
   cms::StreamMessage, 3603  
   decaf::io::DataInput, 1529  
   decaf::io::DataInputStream, 1538  
 readString  
   activemq::commands::ActiveMQBytesMessage, 212  
   activemq::commands::ActiveMQStreamMessage, 516  
   cms::BytesMessage, 1033  
   cms::StreamMessage, 3603  
   decaf::io::DataInput, 1529  
   decaf::io::DataInputStream, 1539  
 readString16  
   activemq::util::MarshallingSupport, 2453  
 readString32  
   activemq::util::MarshallingSupport, 2453  
 Message, 515  
 decaf::io::DataInput, 1530  
 decaf::io::DataInputStream, 1539  
 readUnsignedShort  
   activemq::commands::ActiveMQBytesMessage, 213  
   activemq::commands::ActiveMQStreamMessage, 517  
   cms::BytesMessage, 1033  
   cms::StreamMessage, 3604  
 decaf::io::DataInput, 1530  
 decaf::io::DataInputStream, 1539  
 readUTF  
   activemq::commands::ActiveMQBytesMessage, 213  
   cms::BytesMessage, 1034  
   decaf::io::DataInput, 1531  
   decaf::io::DataInputStream, 1540  
 ready  
   decaf::io::InputStreamReader, 2015  
   decaf::io::Reader, 3113  
 rebalanceConnection  
   activemq::commands::ConnectionControl, 1241  
 RECEIPT  
   activemq::wireformat::stomp::StompCommandConstants, 3576  
 receive  
   activemq::cmsutil::CachedConsumer, 1043  
   activemq::cmsutil::CmsTemplate, 1147,  
   activemq::core::ActiveMQConsumer, 289  
   cms::MessageConsumer, 2552, 2553  
 RECEIVE\_TIMEOUT\_INDEFINITE\_WAIT  
   activemq::cmsutil::CmsTemplate, 1154  
 RECEIVE\_TIMEOUT\_NO\_WAIT  
   activemq::cmsutil::CmsTemplate, 1154  
 ReceiveExecutor  
   activemq::cmsutil::CmsTemplate, 1153  
   activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3120  
 receiveNoWait  
   activemq::cmsutil::CachedConsumer, 1043  
   activemq::core::ActiveMQConsumer, 290  
   cms::MessageConsumer, 2553  
 receiveSelected  
   activemq::cmsutil::CmsTemplate, 1148,  
   1149

- recievedByDFBridge
  - activemq::commands::Message, 2492
- reconnect
  - activemq::transport::failover::FailoverTransport, 1842
  - activemq::transport::IOTransport, 2109
  - activemq::transport::mock::MockTransport, 2730
  - activemq::transport::Transport, 3823
  - activemq::transport::TransportFilter, 3833
- reconnectTo
  - activemq::commands::ConnectionControl, 1241
- recover
  - activemq::cmsutil::PooledSession, 2917
  - activemq::core::ActiveMQSession, 499
  - cms::Session, 3318
- redeliveryCounter
  - activemq::commands::Message, 2492
  - activemq::commands::MessageDispatch, 2559
- RedeliveryPolicy
  - activemq::core::RedeliveryPolicy, 3123
- redispatch
  - activemq::core::ActiveMQSession, 500
- ReentrantLock
  - decaf::util::concurrent::locks::ReentrantLock, 3128
- ReferenceType
  - decaf::lang::ArrayPointer, 699
  - decaf::lang::Pointer, 2898
- registerFactory
  - activemq::transport::TransportRegistry, 3839
  - activemq::wireformat::WireFormatRegistry, 3949
- rejectedExecution
  - decaf::util::concurrent::RejectedExecutionHandler, 3137
- RejectedExecutionException
  - decaf::util::concurrent::RejectedExecutionException, 3134, 3135
- relativize
  - decaf::net::URI, 3862
- release
  - decaf::lang::ArrayPointer, 702
  - decaf::lang::Pointer, 2902
  - decaf::util::concurrent::atomic::AtomicReferenceCounter, 714
  - decaf::util::concurrent::Semaphore, 3286
- releaseAll
  - activemq::cmsutil::ResourceLifecycleManager, 3227
- remainingCapacity
  - decaf::nio::Buffer, 892
- remove
  - activemq::util::ActiveMQProperties, 452
  - cms::CMSProperties, 1137
  - decaf::internal::util::TimerTaskHeap, 3747
  - decaf::util::AbstractCollection, 155
  - decaf::util::AbstractQueue, 166
  - decaf::util::Collection, 1162
  - decaf::util::concurrent::ConcurrentMap, 1200
  - decaf::util::concurrent::ConcurrentStlMap, 1214, 1215
  - decaf::util::concurrent::SynchronousQueue, 3668
  - decaf::util::Iterator, 2115
  - decaf::util::List, 2302
  - decaf::util::Map, 2429
  - decaf::util::PriorityQueue, 2982, 2983
  - decaf::util::Properties, 3079
  - decaf::util::Queue, 3097
  - decaf::util::StlList, 3541
  - decaf::util::StlMap, 3553
  - decaf::util::StlSet, 3570
- removeAll
  - activemq::core::MessageDispatchChannel, 2563
  - decaf::util::AbstractCollection, 156
  - decaf::util::AbstractSet, 169
  - decaf::util::Collection, 1162
  - decaf::util::concurrent::SynchronousQueue, 3668
- removeConsumer
  - activemq::core::ActiveMQSession, 500
  - activemq::state::SessionState, 3379
- removeDispatcher
  - activemq::core::ActiveMQConnection, 258
- removeHandler
  - decaf::util::logging::Logger, 2355
- RemoveInfo
  - activemq::commands::RemoveInfo, 3138

RemoveInfoMarshaller  
     activemq::wireformat::openwire::marshal::RemoveInfoMarshaller, 3154  
     activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 3142  
     activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 3150  
     activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 3162  
     activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 3158  
     activemq::wireformat::openwire::marshal::v6::RemoveInfoMarshaller, 3146  
 removeProducer  
     activemq::core::ActiveMQConnection, 258  
     activemq::core::ActiveMQSession, 500  
     activemq::state::SessionState, 3379  
 removeProperty  
     activemq::wireformat::stomp::StompFrame, 3580  
 removePropertyChangeListener  
     decaf::util::logging::LogManager, 2369  
 removeSession  
     activemq::core::ActiveMQConnection, 258  
     activemq::state::ConnectionState, 1360  
 RemoveSubscriptionInfo  
     activemq::commands::RemoveSubscriptionInfo, 3166  
 RemoveSubscriptionInfoMarshaller  
     activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 3170  
     activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 3179  
     activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 3175  
     activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 3191  
     activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 3187  
     activemq::wireformat::openwire::marshal::v6::RemoveSubscriptionInfoMarshaller, 3183  
 removeSynchronization  
     activemq::core::ActiveMQTransactionCoordinator, 690  
 removeTask  
     activemq::threads::CompositeTaskRunner, 1195  
 removeTempDestination  
     activemq::state::ConnectionState, 1360  
     activemq::state::ConnectionStateTracker, 1367  
     removeTransactionState  
     activemq::state::ConnectionState, 1360  
     removeTransportListener  
     activemq::core::ActiveMQConnection, 258  
     activemq::state::ConnectionState, 1360  
     activemq::transport::CompositeTransport, 1498  
     activemq::transport::failover::FailoverTransport, 1842  
     activemq::transport::failover::URIPool, 3877  
     renegotiateWireFormat  
     activemq::wireformat::openwire::OpenWireFormat, 2845  
     replace  
     decaf::util::concurrent::ConcurrentMap, 1201  
     decaf::util::concurrent::ConcurrentStlMap, 1215, 1216  
     ReplayCommand  
     activemq::commands::ReplayCommand, 3195  
     ReplayCommandMarshaller  
     activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 3202  
     activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 3205  
     activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 3210  
     activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 3215  
     activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 3218  
     activemq::wireformat::openwire::marshal::v6::ReplayCommandMarshaller, 3219  
     replyTo  
     activemq::subscriptions::Message, 2492  
     reportError  
     decaf::util::logging::Handler, 1944  
     request  
     activemq::transport::correlator::ResponseCorrelator, 3235  
     activemq::transport::failover::FailoverTransport, 1843  
     activemq::transport::IOTransport, 2110

- activemq::transport::logging::LoggingTransport, 2362
- activemq::transport::mock::MockTransport, 2730, 2731
- activemq::transport::Transport, 3823, 3824
- activemq::transport::TransportFilter, 3833
- activemq::wireformat::openwire::OpenWireFormat, 2853
- reserve
  - decaf::util::concurrent::ThreadPool, 3725
- reserved
  - z\_stream\_s, 3991
- reset
  - activemq::commands::ActiveMQBytesMessage, 213
  - activemq::commands::ActiveMQStreamMessage, 517
  - activemq::state::ConnectionState, 1360
  - cms::BytesMessage, 1034
  - decaf::internal::util::TimerTaskHeap, 3747
  - decaf::io::BufferedInputStream, 898
  - decaf::io::ByteArrayInputStream, 990
  - decaf::io::ByteArrayOutputStream, 994
  - decaf::io::FilterInputStream, 1859
  - decaf::io::InputStream, 2009
  - decaf::io::PushbackInputStream, 3091
  - decaf::io::Reader, 3114
  - decaf::lang::ArrayPointer, 703
  - decaf::lang::Pointer, 2902
  - decaf::nio::Buffer, 892
  - decaf::util::logging::LogManager, 2369
  - decaf::util::StringTokenizer, 3615
  - decaf::util::zip::Adler32, 692
  - decaf::util::zip::Checksum, 1115
  - decaf::util::zip::CRC32, 1491
  - decaf::util::zip::Deflater, 1677
  - decaf::util::zip::Inflater, 1991
  - decaf::util::zip::InflaterInputStream, 2000
- resize
  - decaf::internal::util::ByteArrayAdapter, 950
- resolve
  - decaf::net::URI, 3863
- resolveDestinationName
  - activemq::cmsutil::CmsDestinationAccessor, 1129
  - activemq::cmsutil::DestinationResolver, 1721
- activemq::cmsutil::DynamicDestinationResolver, 1787
- ResolveProducerExecutor
  - activemq::cmsutil::CmsTemplate, 1153
  - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 3221
- ResolveReceiveExecutor
  - activemq::cmsutil::CmsTemplate, 1153
  - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 3222
- ResourceLifecycleManager
  - activemq::cmsutil::ResourceLifecycleManager, 3225
  - decaf::internal::util::ResourceLifecycleManager, 3224
- Response
  - activemq::commands::Response, 3228
  - ResponseCorrelator
    - activemq::transport::correlator::ResponseCorrelator, 3233
  - ResponseMarshaller
    - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 3256
    - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 3242
    - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 3251
    - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 3237
    - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 3247
    - activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3261
- restore
  - activemq::state::ConnectionStateTracker, 1366
- restoreTransport
  - activemq::transport::failover::FailoverTransport, 1844
- Result
  - activemq::commands::IntegerResponse, 2056
- resume
  - activemq::commands::ConnectionControl, 1241
- retainAll
  - decaf::util::AbstractCollection, 157
  - decaf::util::Collection, 1163
  - decaf::util::concurrent::SynchronousQueue, 3669

- retroactive
  - activemq::commands::ConsumerInfo, 1434
- returnInstance
  - decaf::util::logging::LogWriter, 2376
- returnSession
  - activemq::cmsutil::SessionPool, 3377
- reverse
  - decaf::lang::Integer, 2048
  - decaf::lang::Long, 2387
  - decaf::util::StlQueue, 3562
- reverseBytes
  - decaf::lang::Integer, 2048
  - decaf::lang::Long, 2387
  - decaf::lang::Short, 3387
- rewind
  - decaf::nio::Buffer, 892
- rollback
  - activemq::cmsutil::PooledSession, 2917
  - activemq::core::ActiveMQConsumer, 290
  - activemq::core::ActiveMQSession, 501
  - activemq::core::ActiveMQTransactionContext, 690
  - cms::Session, 3318
- rotateLeft
  - decaf::lang::Integer, 2049
  - decaf::lang::Long, 2387
- rotateRight
  - decaf::lang::Integer, 2049
  - decaf::lang::Long, 2388
- round
  - decaf::lang::Math, 2465, 2466
- run
  - activemq::threads::CompositeTaskRunner, 1196
  - activemq::threads::DedicatedTaskRunner, 1639
  - activemq::transport::inactivity::ReadChecker, 3108
  - activemq::transport::inactivity::WriteChecker, 3951
  - activemq::transport::IOTransport, 2111
  - activemq::transport::mock::InternalCommand, 2086
  - decaf::lang::Runnable, 3265
  - decaf::lang::Thread, 3713
  - decaf::util::concurrent::PooledThread, 2919
- RUNNABLE
  - decaf::lang::Thread, 3710
- RuntimeException
  - decaf::lang::exceptions::RuntimeException, 3268, 3269
- sane
  - inflate\_state, 1984
- schedule
  - decaf::util::Timer, 3733–3738
- scheduleAtFixedRate
  - decaf::util::Timer, 3739–3741
- scheduledExecutionTime
  - decaf::util::TimerTask, 3744
- SECONDS
  - decaf::util::concurrent::TimeUnit, 3757
- SecureRandom
  - decaf::security::SecureRandom, 3271, 3272
- SecureRandomImpl
  - decaf::internal::security::SecureRandomImpl, 3276
- SecureRandomSpi
  - decaf::security::SecureRandomSpi, 3279
- seek
  - gz\_state, 1940
- SEEK\_CUR
  - zconf.h, 4429
- SEEK\_END
  - zconf.h, 4429
- SEEK\_SET
  - zconf.h, 4429
- selector
  - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 3121
  - activemq::commands::ConsumerInfo, 1434
  - activemq::commands::SubscriptionInfo, 3620
- Semaphore
  - decaf::util::concurrent::Semaphore, 3283
- Semaphore
  - decaf::util::concurrent::ConditionHandle, 1227
- SEND
  - activemq::wireformat::stomp::StompCommandConstants, 3576
- send
  - activemq::cmsutil::CachedProducer, 1047–1049
  - activemq::cmsutil::CmsTemplate, 1149, 1150

- activemq::core::ActiveMQProducer, 445–447
- activemq::core::ActiveMQSession, 501
- cms::MessageProducer, 2685–2687
- SendExecutor
  - activemq::cmsutil::CmsTemplate, 1153
  - activemq::cmsutil::CmsTemplate::SendExecutor, 3291
- sendPullRequest
  - activemq::core::ActiveMQConnection, 259
- sendUrgentData
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2817
  - decaf::net::Socket, 3458
  - decaf::net::SocketImpl, 3479
- ServerSocket
  - decaf::net::ServerSocket, 3294, 3295
  - decaf::net::Socket, 3463
- ServerSocketFactory
  - decaf::net::ServerSocketFactory, 3302
- serviceName
  - activemq::commands::DiscoveryEvent, 1725
- SESSION\_TRANSACTED
  - cms::Session, 3308
- SessionId
  - activemq::commands::SessionId, 3321, 3322
- sessionId
  - activemq::commands::ConsumerId, 1401
  - activemq::commands::ProducerId, 3018
  - activemq::commands::SessionInfo, 3351
- SessionIdMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 3345
  - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 3325
  - activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 3341
  - activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 3329
  - activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 3337
  - activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3333
- SessionInfo
  - activemq::commands::SessionInfo, 3349
- SessionInfoMarshaller
  - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 3361
  - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 3369
  - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 3365
  - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 3373
  - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 3357
  - activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3353
- SessionPool
  - activemq::cmsutil::SessionPool, 3376
- SessionState
  - activemq::state::SessionState, 3378
- set
  - decaf::util::concurrent::atomic::AtomicBoolean, 707
  - decaf::util::concurrent::atomic::AtomicInteger, 712
  - decaf::util::concurrent::atomic::AtomicReference, 717
  - decaf::util::List, 2302
  - decaf::util::ListIterator, 2306
  - decaf::util::StillList, 3542
- setAbsolute
  - decaf::internal::net::URIType, 3888
- setAckHandler
  - activemq::commands::Message, 2487
- setAckMode
  - activemq::commands::SessionInfo, 3351
- setAckType
  - activemq::commands::MessageAck, 2524
- setAdditionalPredicate
  - activemq::commands::ConsumerInfo, 1431
- setAdviser
  - activemq::commands::ActiveMQDestination, 3311
- setAlwaysSyncSend
  - activemq::core::ActiveMQConnection, 259
- setAlwaysSyncSend
  - activemq::core::ActiveMQConnectionFactory, 271
- setArrive
  - activemq::commands::Message, 2487
- setAuthority
  - decaf::internal::net::URIType, 3888
- setBackOffMultiplier

activemq::core::policies::DefaultRedeliveryPolicy, 1647  
 activemq::core::RedeliveryPolicy, 3125  
 activemq::transport::failover::FailoverTransport, 1844  
 setBackup, 1844  
 activemq::transport::failover::FailoverTransport, 1844  
 setBackupPoolSize, 722  
 activemq::transport::failover::BackupTransport, 1844  
 activemq::transport::failover::FailoverTransport, 1844  
 setBlockSize, 3723  
 decaf::util::concurrent::ThreadPool, 3723  
 setBody, 3580  
 activemq::wireformat::stomp::StompFrame, 3580  
 setBodyBytes, 214  
 activemq::commands::ActiveMQBytesMessage, 214  
 cms::BytesMessage, 1034  
 setBool, 2936  
 activemq::util::PrimitiveList, 2936  
 activemq::util::PrimitiveMap, 2948  
 activemq::util::PrimitiveValueNode, 2971  
 setBoolean, 340  
 activemq::commands::ActiveMQMapMessage, 340  
 cms::MapMessage, 2438  
 setBooleanProperty, 407  
 activemq::commands::ActiveMQMessageProperty, 407  
 activemq::wireformat::openwire::utils::MessageProperty, 2693  
 cms::Message, 2510  
 setBranchQualifier, 3963  
 activemq::commands::XATransactionId, 3963  
 setBrokerId, 860  
 activemq::commands::BrokerInfo, 860  
 setBrokerInTime, 2487  
 activemq::commands::Message, 2487  
 setBrokerMasterConnector, 1328  
 activemq::commands::ConnectionInfo, 1328  
 setBrokerName, 860  
 activemq::commands::BrokerInfo, 860  
 activemq::commands::DiscoveryEvent, 1724  
 setBrokerOutTime, 1328  
 activemq::commands::Message, 2487  
 setBrokerPath, 1328  
 activemq::commands::ConnectionInfo, 1328  
 activemq::commands::ConsumerInfo, 1431  
 activemq::commands::DestinationInfo, 1694  
 activemq::commands::Message, 2487  
 activemq::commands::ProducerInfo, 3046  
 setBrokerSequenceId, 2627  
 activemq::commands::MessageId, 2627  
 setBrokerUploadUrl, 860  
 activemq::commands::BrokerInfo, 860  
 setBrokerURL, 259  
 activemq::core::ActiveMQConnection, 272  
 activemq::core::ActiveMQConnectionFactory, 272  
 setBrowser, 1431  
 activemq::commands::ConsumerInfo, 1431  
 setByte, 340  
 activemq::commands::ActiveMQMapMessage, 340  
 activemq::util::PrimitiveList, 2937  
 activemq::util::PrimitiveMap, 2948  
 activemq::util::PrimitiveValueNode, 2971  
 cms::MapMessage, 2439  
 setByteArray, 2937  
 activemq::util::PrimitiveList, 2937  
 activemq::util::PrimitiveMap, 2948  
 activemq::util::PrimitiveValueNode, 2972  
 decaf::io::BlockingByteArrayInputStream, 803  
 decaf::io::ByteArrayInputStream, 990, 991  
 setByteProperty, 408  
 activemq::commands::ActiveMQMessageTemplate, 408  
 activemq::wireformat::openwire::utils::MessagePropertyInterface, 2693  
 cms::Message, 2511  
 setBytes, 340  
 activemq::commands::ActiveMQMapMessage, 340  
 cms::MapMessage, 2439  
 setCacheEnabled



activemq::commands::WireFormatInfo,	activemq::core::ActiveMQConnectionFactory,
3919	272
activemq::wireformat::openwire::OpenWireFormat,	activemq::commands::Message, 2488
2845	setCMSCorrelationID
setCacheSize	activemq::commands::ActiveMQMessageTemplate,
activemq::commands::WireFormatInfo,	408
3919	activemq::commands::Message, 2511
activemq::wireformat::openwire::OpenWireFormat,	setCMSDeliveryMode
2845	activemq::commands::ActiveMQMessageTemplate,
setCause	408
activemq::commands::BrokerError, 826	cms::Message, 2512
setChar	setCMSDestination
activemq::commands::ActiveMQMapMessage,	activemq::commands::ActiveMQMessageTemplate,
341	408
activemq::util::PrimitiveList, 2938	cms::Message, 2513
activemq::util::PrimitiveMap, 2949	setCMSExpiration
activemq::util::PrimitiveValueNode, 2972	activemq::commands::ActiveMQMessageTemplate,
cms::MapMessage, 2440	409
setCipherSuites	cms::Message, 2513
decaf::net::ssl::SSLParameters, 3497	setCMSMessageID
setClientID	activemq::commands::ActiveMQMessageTemplate,
activemq::core::ActiveMQConnection,	409
259	cms::Message, 2514
cms::Connection, 1236	setCMSPriority
setClientId	activemq::commands::ActiveMQMessageTemplate,
activemq::commands::ConnectionInfo,	409
1329	cms::Message, 2514
activemq::commands::JournalTopicAck,	setCMSRedelivered
2146	activemq::commands::ActiveMQMessageTemplate,
activemq::commands::RemoveSubscriptionInfo,	410
3168	cms::Message, 2515
activemq::commands::SubscriptionInfo,	setCMSReplyTo
3619	activemq::commands::ActiveMQMessageTemplate,
activemq::core::ActiveMQConnectionFactory,	410
272	cms::Message, 2515
setClientMaster	setCMSTimestamp
activemq::commands::ConnectionInfo,	activemq::commands::ActiveMQMessageTemplate,
1329	410
setClose	cms::Message, 2516
activemq::commands::ConnectionControl,	setCMSType
1240	activemq::commands::ActiveMQMessageTemplate,
activemq::commands::ConsumerControl,	411
1372	cms::Message, 2516
setClosed	setCollisionAvoidancePercent
activemq::transport::failover::BackupTransport,	activemq::core::policies::DefaultRedeliveryPolicy,
720	1647
setCloseTimeout	activemq::core::RedeliveryPolicy, 3125
activemq::core::ActiveMQConnection,	setCommand
260	

activemq::commands::ControlCommand, 1461  
 activemq::wireformat::stomp::StompFrame, 3580  
 setCommandId  
   activemq::commands::BaseCommand, 729  
   activemq::commands::Command, 1169  
   activemq::commands::PartialCommand, 2869  
 setComponents  
   activemq::util::CompositeData, 1192  
 setCompressed  
   activemq::commands::Message, 2488  
 setConnectedBrokers  
   activemq::commands::ConnectionControl, 1240  
 setConnection  
   activemq::commands::ActiveMQTempDestination, 550  
   activemq::commands::Message, 2488  
 setConnectionFactory  
   activemq::cmsutil::CmsAccessor, 1127  
 setConnectionId  
   activemq::commands::BrokerInfo, 860  
   activemq::commands::ConnectionError, 1268  
   activemq::commands::ConnectionInfo, 1329  
   activemq::commands::ConsumerId, 1401  
   activemq::commands::DestinationInfo, 1694  
   activemq::commands::LocalTransactionId, 2309  
   activemq::commands::ProducerId, 3018  
   activemq::commands::RemoveSubscriptionInfo, 3168  
   activemq::commands::SessionId, 3323  
   activemq::commands::TransactionInfo, 3788  
 setConnectionInterruptProcessingComplete  
   activemq::state::ConnectionState, 1360  
   activemq::transport::failover::FailoverTransport, 1844  
 setConsumerId  
   activemq::commands::ConsumerControl, 1372  
   activemq::commands::ConsumerInfo, 1431  
   activemq::commands::MessageAck, 2524  
   activemq::commands::MessageDispatch, 2558  
   activemq::commands::MessageDispatchNotification, 2593  
   activemq::commands::MessagePull, 2698  
   activemq::commands::Message, 2488  
   activemq::commands::MessagePull, 2698  
   activemq::commands::Response, 3230  
   setData  
     activemq::commands::DataArrayResponse, 1495  
     activemq::commands::DataResponse, 1552  
     activemq::commands::PartialCommand, 2869  
   setDataStructure  
     activemq::commands::Message, 2488  
   setDefault  
     decaf::net::ssl::SSLContext, 3491  
   setDefaultClientId  
     activemq::core::ActiveMQConnection, 260  
   setDefaultDestination  
     activemq::cmsutil::CmsTemplate, 1150  
   setDefaultDestinationName  
     activemq::cmsutil::CmsTemplate, 1151  
   setDeletedByBroker  
     activemq::commands::ActiveMQBlobMessage, 176  
   setDeliveryMode  
     activemq::cmsutil::CachedProducer, 1050  
     activemq::cmsutil::CmsTemplate, 1151  
     activemq::core::ActiveMQProducer, 447  
     cms::MessageProducer, 2687  
   setDeliveryPersistent  
     activemq::cmsutil::CmsTemplate, 1151  
   setDeliverySequenceId  
     activemq::commands::MessageDispatchNotification, 2594  
   setDestination  
     activemq::commands::ConsumerControl, 1372  
     activemq::commands::ConsumerInfo, 1431  
     activemq::commands::DestinationInfo, 1695

activemq::commands::JournalQueueAck, 2118  
 activemq::commands::JournalTopicAck, 2146  
 activemq::commands::Message, 2488  
 activemq::commands::MessageAck, 2524  
 activemq::commands::MessageDispatch, 2558  
 activemq::commands::MessageDispatchNotification, 2594  
 activemq::commands::MessagePull, 2698  
 activemq::commands::ProducerInfo, 3046  
 activemq::commands::SubscriptionInfo, 3619  
 setDestinationResolver  
   activemq::cmsutil::CmsDestinationAccessor, 1130  
 setDictionary  
   decaf::util::zip::Deflater, 1677, 1678  
   decaf::util::zip::Inflater, 1991, 1992  
 setDisableMessageID  
   activemq::cmsutil::CachedProducer, 1050  
   activemq::core::ActiveMQProducer, 447  
   cms::MessageProducer, 2688  
 setDisableMessageTimeStamp  
   activemq::cmsutil::CachedProducer, 1050  
   activemq::core::ActiveMQProducer, 448  
   cms::MessageProducer, 2688  
 setDispatchAsync  
   activemq::commands::ConsumerInfo, 1432  
   activemq::commands::ProducerInfo, 3046  
   activemq::core::ActiveMQConnection, 260  
   activemq::core::ActiveMQConnectionFactory, 272  
 setDouble  
   activemq::commands::ActiveMQMapMessage, 341  
   activemq::util::PrimitiveList, 2938  
   activemq::util::PrimitiveMap, 2949  
   activemq::util::PrimitiveValueNode, 2972  
   cms::MapMessage, 2440  
 setDoubleProperty  
   activemq::commands::ActiveMQMessageTemplate, 411  
   activemq::wireformat::openwire::utils::MessageProperty, 2694  
   cms::Message, 2517  
 setDroppable  
   activemq::commands::Message, 2488  
   setDuplexConnection  
   activemq::commands::BrokerInfo, 860  
   setDurableTopicPrefetch  
   activemq::core::policies::DefaultPrefetchPolicy, 1642  
   activemq::core::PrefetchPolicy, 2927  
   setEnabled  
   activemq::transport::failover::BackupTransportPool, 723  
   setEnabledCipherSuites  
   decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796  
   decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2801  
   decaf::internal::net::ssl::openssl::OpenSSLSocket, 2818  
   decaf::net::ssl::SSLServerSocket, 3502  
   decaf::net::ssl::SSLSocket, 3512  
   setEnabledProtocols  
   decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796  
   decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2802  
   decaf::internal::net::ssl::openssl::OpenSSLSocket, 2818  
   decaf::net::ssl::SSLServerSocket, 3503  
   decaf::net::ssl::SSLSocket, 3512  
   setenv  
   decaf::lang::System, 3677  
   setErrorHandler  
   decaf::util::logging::Handler, 1944  
   setException  
   activemq::commands::ConnectionError, 1268  
   activemq::commands::ExceptionResponse, 1804  
   setExceptionHandler  
   activemq::commands::BrokerError, 826  
   setExceptionHandler  
   activemq::core::ActiveMQConnection, 261  
   activemq::core::ActiveMQConnectionFactory, 272  
   updateConnection, 1236  
   setExclusive  
   getPropertyInterceptors, ActiveMQDestination, 301  
   activemq::commands::ConsumerInfo, 1432

setExit  
     activemq::commands::ConnectionControl, 1240  
 setExpiration  
     activemq::commands::Message, 2488  
 setExplicitQosEnabled  
     activemq::cmsutil::CmsTemplate, 1152  
 setFailOnClose  
     activemq::transport::mock::MockTransport, 2731  
 setFailOnKeepAliveSends  
     activemq::transport::mock::MockTransport, 2731  
 setFailOnReceiveMessage  
     activemq::transport::mock::MockTransport, 2731  
 setFailOnSendMessage  
     activemq::transport::mock::MockTransport, 2731  
 setFailOnStart  
     activemq::transport::mock::MockTransport, 2731  
 setFailOnStop  
     activemq::transport::mock::MockTransport, 2731  
 setFaultTolerant  
     activemq::commands::ConnectionControl, 1240  
     activemq::commands::ConnectionInfo, 1329  
 setFaultTolerantConfiguration  
     activemq::commands::BrokerInfo, 860  
 setFilter  
     decaf::util::logging::Handler, 1944  
     decaf::util::logging::Logger, 2355  
 setFirstMessageld  
     activemq::commands::MessageAck, 2524  
 setFirstNakNumber  
     activemq::commands::ReplayCommand, 3196  
 setFloat  
     activemq::commands::ActiveMQMapMessage, 342  
     activemq::util::PrimitiveList, 2938  
     activemq::util::PrimitiveMap, 2949  
     activemq::util::PrimitiveValueNode, 2972  
     cms::MapMessage, 2441  
 setFloatProperty  
     activemq::commands::ActiveMQMessageTemplate, 411  
     activemq::wireformat::openwire::utils::MessagePropertyInterce, 2694  
     cms::Message, 2518  
 setFlush  
     activemq::commands::ConsumerControl, 1372  
 setFormatId  
     activemq::commands::XATransactionId, 3963  
 setFormatter  
     decaf::util::logging::Handler, 1944  
 setFragment  
     activemq::util::CompositeData, 1192  
 setGlobalTransactionId  
     decaf::internal::net::URIType, 3888  
     activemq::commands::XATransactionId, 3963  
 setGroupId  
     activemq::commands::Message, 2488  
 setGroupSequence  
     activemq::commands::Message, 2488  
 setHost  
     activemq::util::CompositeData, 1192  
     decaf::internal::net::URIType, 3889  
 setInitialDelayTime  
     activemq::transport::inactivity::InactivityMonitor, 1967  
 setInitialized  
     activemq::transport::failover::FailoverTransport, 1844  
 setInitialReconnectDelay  
     activemq::transport::failover::FailoverTransport, 1844  
 setInitialRedeliveryDelay  
     activemq::core::policies::DefaultRedeliveryPolicy, 1647  
     activemq::core::RedeliveryPolicy, 3125  
 setInput  
     decaf::util::zip::Deflater, 1679  
     decaf::util::zip::Inflater, 1992, 1993  
 setInputStream  
     activemq::transport::IOTransport, 2111  
 setInt  
     activemq::commands::ActiveMQMapMessage, 342  
     activemq::util::PrimitiveList, 2939  
     activemq::util::PrimitiveMap, 2949  
     activemq::util::PrimitiveValueNode, 2972  
     cms::MapMessage, 2441  
 setIntProperty

activemq::commands::ActiveMQMessageTemplate, 412  
 activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2694  
 cms::Message, 2518  
 setKeepAlive  
   decaf::net::Socket, 3459  
 setKeepAliveResponseRequired  
   activemq::transport::inactivity::InactivityMonitor, 1967  
 setLastDeliveredSequenceId  
   activemq::commands::RemoveInfo, 3140  
   activemq::core::ActiveMQConsumer, 2969  
   activemq::core::ActiveMQSession, 501  
 setLastMessageId  
   activemq::commands::MessageAck, 2524  
 setLastNakNumber  
   activemq::commands::ReplayCommand, 3196  
 setLevel  
   decaf::util::logging::Handler, 1945  
   decaf::util::logging::Logger, 2355  
   decaf::util::logging::LogRecord, 2373  
   decaf::util::zip::Deflater, 1680  
 setLimit  
   activemq::util::MemoryUsage, 2474  
 setList  
   activemq::util::PrimitiveValueNode, 2973  
 setLoggerName  
   decaf::util::logging::LogRecord, 2373  
 setLong  
   activemq::commands::ActiveMQMapMessage, 342  
   activemq::util::PrimitiveList, 2939  
   activemq::util::PrimitiveMap, 2950  
   activemq::util::PrimitiveValueNode, 2973  
   cms::MapMessage, 2441  
 setLongProperty  
   activemq::commands::ActiveMQMessageTemplate, 412  
   activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2694  
   cms::Message, 2519  
 setMagic  
   activemq::commands::WireFormatInfo, 3919  
 setManageable  
   activemq::commands::ConnectionInfo, 1329  
 setManaged  
   activemq::internal::util::GenericResource, 1938  
   setAppPropertyInterceptor, 2694  
   activemq::util::PrimitiveValueNode, 2973  
   setMark  
     cms::CMSException, 1133  
     decaf::lang::Exception, 1799  
     decaf::lang::Throwable, 3727  
   setMarshalledForm  
     activemq::commands::BaseDataStructure, 796  
     activemq::wireformat::MarshalAware, 2446  
   setMarshalledProperties  
     activemq::commands::Message, 2488  
     activemq::commands::WireFormatInfo, 3920  
   setMasterBroker  
     activemq::commands::BrokerInfo, 860  
   setMaxCacheSize  
     activemq::state::ConnectionStateTracker, 1366  
     activemq::transport::failover::FailoverTransport, 1844  
   setMaximumPendingMessageLimit  
     activemq::commands::ConsumerInfo, 1432  
   setMaximumRedeliveries  
     activemq::core::policies::DefaultRedeliveryPolicy, 1647  
     activemq::core::RedeliveryPolicy, 3126  
   setMaxInactivityDuration  
     activemq::commands::WireFormatInfo, 3920  
     activemq::wireformat::openwire::OpenWireFormat, 2845  
   setMaxInactivityDurationInitialDelay  
     activemq::commands::WireFormatInfo, 3920  
   setMaxReconnectAttempts  
     activemq::transport::failover::FailoverTransport, 1845  
   setMaxReconnectDelay  
     activemq::transport::failover::FailoverTransport, 1845  
   setMaxThreads  
     decaf::util::concurrent::ThreadPool, 3723  
   setMessage

activemq::commands::BrokerError, 826  
 activemq::commands::JournalTrace, 2174  
 activemq::commands::MessageDispatch, 2558  
 decaf::lang::Exception, 1800  
 decaf::util::logging::LogRecord, 2373  
 setMessageAck  
 activemq::commands::JournalQueueAck, 2118  
 setMessageCount  
 activemq::commands::MessageAck, 2525  
 setMessageId  
 activemq::commands::JournalTopicAck, 2146  
 activemq::commands::Message, 2488  
 activemq::commands::MessageDispatchNotification, 2594  
 activemq::commands::MessagePull, 2639  
 setMessageIdEnabled  
 activemq::cmsutil::CmsTemplate, 1152  
 setMessageListener  
 activemq::cmsutil::CachedConsumer, 1044  
 activemq::core::ActiveMQConsumer, 294  
 cms::MessageConsumer, 2553  
 setMessageSequenceId  
 activemq::commands::JournalTopicAck, 2146  
 setMessageTimestampEnabled  
 activemq::cmsutil::CmsTemplate, 1152  
 setMimeType  
 activemq::commands::ActiveMQBlobMessage, 176  
 setName  
 activemq::commands::ActiveMQBlobMessage, 176  
 decaf::lang::Thread, 3713  
 setNeedClientAuth  
 decaf::internal::net::ssl::openssl::OpenSSLParameters, 2796  
 decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 2802  
 decaf::internal::net::ssl::openssl::OpenSSLSSLSocket, 2818  
 decaf::net::ssl::SSLParameters, 3497  
 decaf::net::ssl::SSLServerSocket, 3503  
 decaf::net::ssl::SSLSocket, 3513  
 setNetworkBrokerId  
 activemq::commands::NetworkBridgeFilter, 2748  
 setNetworkConnection  
 activemq::commands::BrokerInfo, 861  
 setNetworkConsumerPath  
 activemq::commands::ConsumerInfo, 1432  
 setNetworkProperties  
 activemq::commands::BrokerInfo, 861  
 setNetworkSubscription  
 activemq::commands::ConsumerInfo, 1432  
 setNetworkTTL  
 activemq::commands::NetworkBridgeFilter, 2748  
 setNoLocal  
 activemq::cmsutil::CmsTemplate, 1152  
 activemq::commands::ConsumerInfo, 1432  
 setNoRangeAcks  
 activemq::commands::ConsumerInfo, 1432  
 setNumReceivedMessageBeforeFail  
 activemq::transport::mock::MockTransport, 2731  
 setNumReceivedMessages  
 activemq::transport::mock::MockTransport, 2732  
 setNumSentKeepAlives  
 activemq::transport::mock::MockTransport, 2732  
 setNumSentKeepAlivesBeforeFail  
 activemq::transport::mock::MockTransport, 2732  
 setNumSentMessageBeforeFail  
 activemq::transport::mock::MockTransport, 2732  
 setNumSentMessages  
 activemq::transport::mock::MockTransport, 2732  
 setOperationType  
 activemq::commands::DestinationInfo, 1695  
 setOptimizedAcknowledge

- activemq::commands::ConsumerInfo, 1432
- activemq::wireformat::openwire::OpenWireFormat, 2846
- setOption
  - decaf::internal::net::tcp::TcpSocket, 3689
  - decaf::net::SocketImpl, 3479
- setPrefetch
  - activemq::commands::ConsumerControl, 1372
- setPrefetchPolicy
  - activemq::core::ActiveMQConnection, 261
- setOrdered
  - activemq::commands::ActiveMQDestination, 301
- setOrderedTarget
  - activemq::commands::ActiveMQDestination, 302
  - activemq::core::ActiveMQConnectionFactory, 273
- setOriginalDestination
  - activemq::commands::Message, 2489
- setOriginalTransactionId
  - activemq::commands::Message, 2489
- setPrefetchSize
  - activemq::commands::ConsumerInfo, 1432
- setPrepared
  - activemq::state::TransactionState, 3814
- setPreparedResult
  - activemq::state::TransactionState, 3814
- setOutputStream
  - decaf::util::logging::StreamHandler, 3594
- setPriority
  - activemq::cmsutil::CachedProducer, 1051
  - activemq::cmsutil::CmsTemplate, 1152
  - activemq::commands::ConsumerInfo, 1432
  - activemq::commands::Message, 2489
  - activemq::core::ActiveMQProducer, 448
  - cms::MessageProducer, 2688
  - decaf::lang::Thread, 3713
- setOutgoingListener
  - activemq::transport::mock::MockTransport, 2732
- setProducerId
  - activemq::commands::Message, 2489
  - activemq::commands::MessageId, 2627
  - activemq::commands::ProducerAck, 2986
  - activemq::commands::ProducerInfo, 3046
- setProducerSequenceId
  - activemq::commands::MessageId, 2627
- setProducerSessionKey
  - activemq::commands::ProducerId, 3018
- setProducerWindowSize
  - activemq::core::ActiveMQConnection, 261
  - activemq::core::ActiveMQConnectionFactory, 273
- setPath
  - activemq::util::CompositeData, 1192
  - decaf::internal::net::URIType, 3889
- setPeerBrokerInfos
  - activemq::commands::BrokerInfo, 861
- setPersistent
  - activemq::commands::Message, 2489
- setPhysicalName
  - activemq::commands::ActiveMQDestination, 302
- setPooledThreadListener
  - decaf::util::concurrent::PooledThread, 2920
- setProperty
  - activemq::util::ActiveMQProperties, 452
  - decaf::util::logging::LogManager, 2369
  - activemq::util::ActiveMQProperties, 452
  - activemq::wireformat::stomp::StompFrame, 3580
  - cms::CMSProperties, 1138
- setPort
  - decaf::internal::net::URIType, 3889
- setPreferredWireFormatInfo

- decaf::lang::System, 3677
- decaf::util::Properties, 3079
- setProtocols
  - decaf::net::ssl::SSLParameters, 3497
- setPubSubDomain
  - activemq::cmsutil::CmsDestinationAccessor, 1240
  - 1130
  - activemq::cmsutil::CmsTemplate, 1152
- setQuery
  - decaf::internal::net::URIType, 3889
- setQueueBrowserPrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1643
  - activemq::core::PrefetchPolicy, 2928
- setQueuePrefetch
  - activemq::core::policies::DefaultPrefetchPolicy, 1643
  - activemq::core::PrefetchPolicy, 2928
- setRandomize
  - activemq::transport::failover::FailoverTransport, 1845
  - activemq::transport::failover::URIPool, 3877
- setReadCheckTime
  - activemq::transport::inactivity::InactivityMonitor, 1967
- setReadOnly
  - decaf::internal::nio::ByteBuffer, 983
  - decaf::internal::nio::CharArrayBuffer, 1087
  - decaf::internal::nio::DoubleArrayBuffer, 1772
  - decaf::internal::nio::FloatArrayBuffer, 1886
  - decaf::internal::nio::IntArrayBuffer, 2025
  - decaf::internal::nio::LongArrayBuffer, 2402
  - decaf::internal::nio::ShortArrayBuffer, 3400
- setReadOnlyBody
  - activemq::commands::Message, 2489
- setReadOnlyProperties
  - activemq::commands::Message, 2489
- setRebalanceConnection
  - activemq::commands::ConnectionControl, 1240
- setReceiveBufferSize
  - decaf::net::ServerSocket, 3299
  - decaf::net::Socket, 3459
- setReceiveTimeout
  - activemq::cmsutil::CmsTemplate, 1153
- setRecievedByDFBridge
  - activemq::commands::Message, 2489
- setReconnectDelay
  - activemq::transport::failover::FailoverTransport, 1845
- setReconnectTo
  - activemq::commands::ConnectionControl, 1240
- setRedeliveryCounter
  - activemq::commands::Message, 2489
- setRedeliveryPolicy
  - activemq::core::ActiveMQConnection, 262
  - activemq::core::ActiveMQConnectionFactory, 273
- setRemoteBlobUrl
  - activemq::core::ActiveMQConsumer, 291
- setReplyTo
  - activemq::commands::ActiveMQBlobMessage, 176
- setResponse
  - activemq::transport::correlator::FutureResponse, 1934
- setResponseBuilder
  - activemq::transport::mock::InternalCommandListener, 2086
  - activemq::transport::mock::MockTransport, 2732
- setResponseRequired
  - activemq::commands::BaseCommand, 729
  - activemq::commands::Command, 1169
- setRestoreConsumers
  - activemq::state::ConnectionStateTracker, 1366
- setRestoreProducers
  - activemq::state::ConnectionStateTracker, 1366
- setRestoreSessions
  - activemq::state::ConnectionStateTracker, 1366
- setRestoreTransaction
  - activemq::state::ConnectionStateTracker, 1366
- setResult
  - activemq::commands::IntegerResponse, 2056
- setResume



- activemq::commands::ConnectionControl, 1240
- setRetroactive
  - activemq::commands::ConsumerInfo, 1432
- setReuseAddress
  - decaf::net::ServerSocket, 3300
  - decaf::net::Socket, 3460
- setScheduledTime
  - decaf::util::TimerTask, 3744
- setScheme
  - activemq::util::CompositeData, 1192
  - decaf::internal::net::URIType, 3890
- setSchemeSpecificPart
  - decaf::internal::net::URIType, 3890
- setSeed
  - decaf::security::SecureRandom, 3274
  - decaf::util::Random, 3105
- setSelector
  - activemq::commands::ConsumerInfo, 1432
  - activemq::commands::SubscriptionInfo, 3619
- setSendBufferSize
  - decaf::net::Socket, 3460
- setSendTimeout
  - activemq::core::ActiveMQConnection, 262
  - activemq::core::ActiveMQConnectionFactory, 274
  - activemq::core::ActiveMQProducer, 448
- setServerAuthority
  - decaf::internal::net::URIType, 3890
- setServiceName
  - activemq::commands::DiscoveryEvent, 1724
- setSessionAcknowledgeMode
  - activemq::cmsutil::CmsAccessor, 1127
- setSessionId
  - activemq::commands::ConsumerId, 1401
  - activemq::commands::ProducerId, 3018
  - activemq::commands::SessionInfo, 3351
- setShort
  - activemq::commands::ActiveMQMapMessage, 343
  - activemq::util::PrimitiveList, 2939
  - activemq::util::PrimitiveMap, 2950
  - activemq::util::PrimitiveValueNode, 2973
  - cms::MapMessage, 2442
- setShortProperty
- activemq::commands::ActiveMQMessageTemplate, 413
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2694
- cms::Message, 2519
- setSize
  - activemq::commands::ProducerAck, 2987
- setSizePrefixDisabled
  - activemq::commands::WireFormatInfo, 3921
  - activemq::wireformat::openwire::OpenWireFormat, 2846
- setSlaveBroker
  - activemq::commands::BrokerInfo, 861
- setSocketImplFactory
  - decaf::net::ServerSocket, 3300
  - decaf::net::Socket, 3460
- setSoLinger
  - decaf::net::Socket, 3461
- setSoTimeout
  - decaf::net::ServerSocket, 3300
  - decaf::net::Socket, 3461
- setSource
  - decaf::internal::net::URIType, 3890
- setSourceFile
  - decaf::util::logging::LogRecord, 2374
- setSourceFunction
  - decaf::util::logging::LogRecord, 2374
- setSourceLine
  - decaf::util::logging::LogRecord, 2374
- setSSLParameters
  - decaf::net::ssl::SSLSocket, 3513
- setStackTrace
  - decaf::lang::Exception, 1800
- setStackTraceElements
  - activemq::commands::BrokerError, 827
- setStackTraceEnabled
  - activemq::commands::WireFormatInfo, 3921
- activemq::wireformat::openwire::OpenWireFormat, 2846
- setStart
  - activemq::commands::ConsumerControl, 1372
- setStartupMaxReconnectAttempts
  - activemq::transport::failover::FailoverTransport, 1845
- setStop
  - activemq::commands::ConsumerControl, 1372

setStrategy  
     decaf::util::zip::Deflater, 1680  
 setString  
     activemq::commands::ActiveMQMapMessage, 3921  
         343  
     activemq::util::PrimitiveList, 2940  
     activemq::util::PrimitiveMap, 2950  
     activemq::util::PrimitiveValueNode, 2974  
     cms::MapMessage, 2442  
 setStringProperty  
     activemq::commands::ActiveMQMessageTemplate, 695  
         413  
     activemq::wireformat::openwire::utils::MessageProperty, 1845  
         2695  
     cms::Message, 2520  
 setSubscriptionName  
     activemq::commands::RemoveSubscriptionInfo, 3168  
     activemq::commands::SubscriptionInfo, 3619  
 setSubscribedDestination  
     activemq::commands::SubscriptionInfo, 3619  
 setSubscriptionName  
     activemq::commands::ConsumerInfo, 1432  
 setSubscriptionName  
     activemq::commands::JournalTopicAck, 2146  
 setSuspend  
     activemq::commands::ConnectionControl, 1240  
 setSynchronizationRegistered  
     activemq::core::ActiveMQConsumer, 291  
 setTargetConsumerId  
     activemq::commands::Message, 2489  
 setTcpNoDelay  
     decaf::net::Socket, 3461  
 setTcpNoDelayEnabled  
     activemq::commands::WireFormatInfo, 3921  
     activemq::wireformat::openwire::OpenWireFormat, 2847  
 setText  
     activemq::commands::ActiveMQTextMessage, 634  
     cms::TextMessage, 3706  
 setTextView  
     activemq::commands::MessageId, 2627  
 setThrown  
     decaf::util::logging::LogRecord, 2374  
     setTightEncodingEnabled  
         activemq::commands::WireFormatInfo, 3921  
         activemq::wireformat::openwire::OpenWireFormat, 2847  
     setTime  
         decaf::util::Date, 1636  
     setTimeout  
         activemq::commands::DestinationInfo, 695  
         activemq::commands::MessagePull, 2698  
         activemq::wireformat::openwire::utils::MessageProperty, 1845  
         activemq::transport::failover::FailoverTransport, 1845  
     setTimestamp  
         activemq::commands::Message, 2490  
     decaf::util::logging::LogRecord, 2375  
     setTimeToLive  
         activemq::cmsutil::CachedProducer, 1051  
         activemq::cmsutil::CmsTemplate, 1153  
         activemq::core::ActiveMQProducer, 448  
         cms::MessageProducer, 2689  
     setTopicPrefetch  
         activemq::core::policies::DefaultPrefetchPolicy, 1643  
         activemq::core::PrefetchPolicy, 2928  
     setTrackMessages  
         activemq::state::ConnectionStateTracker, 1366  
         activemq::transport::failover::FailoverTransport, 1845  
     setTrackTransactionProducers  
         activemq::state::ConnectionStateTracker, 1367  
     activemq::transport::failover::FailoverTransport, 1845  
     setTrackTransactions  
         activemq::state::ConnectionStateTracker, 1367  
     setTrafficClass  
         decaf::net::Socket, 3462  
     setTransactionId  
         activemq::commands::JournalTopicAck, 2147  
         activemq::commands::JournalTransaction, 2201  
         activemq::commands::Message, 2490  
         activemq::commands::MessageAck, 2525  
         activemq::commands::TransactionInfo, 3788

- setTransactionState
  - activemq::state::ProducerState, 3072
- setTransport
  - activemq::transport::failover::BackupTransport, 720
  - activemq::transport::mock::InternalCommandListener, 2086
- setTransportInterruptionProcessingComplete
  - activemq::core::ActiveMQConnection, 262
- setTransportListener
  - activemq::transport::failover::FailoverTransport, 1845
  - activemq::transport::IOTransport, 2111
  - activemq::transport::mock::MockTransport, 2732
  - activemq::transport::Transport, 3824
  - activemq::transport::TransportFilter, 3834
- setTreadId
  - decaf::util::logging::LogRecord, 2375
- setType
  - activemq::commands::JournalTransaction, 2201
  - activemq::commands::Message, 2490
  - activemq::commands::TransactionInfo, 3788
- setUncaughtExceptionHandler
  - decaf::lang::Thread, 3714
- setupSocketImpl
  - decaf::net::ServerSocket, 3301
- setUri
  - activemq::transport::failover::BackupTransport, 720
- setUsage
  - activemq::util::MemoryUsage, 2475
- setUseAsyncSend
  - activemq::core::ActiveMQConnection, 262
  - activemq::core::ActiveMQConnectionFactory, 274
- setUseClientMode
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2797
  - decaf::internal::net::ssl::openssl::OpenSSLSocket, 2819
  - decaf::net::ssl::SSLSocket, 3514
- setUseCollisionAvoidance
  - activemq::core::policies::DefaultRedeliveryPolicy, 1648
  - activemq::core::RedeliveryPolicy, 3126
- setUseCompression
  - activemq::core::ActiveMQConnection, 263
  - activemq::core::ActiveMQConnectionFactory, 274
- setUseExponentialBackOff
  - activemq::core::policies::DefaultRedeliveryPolicy, 1648
  - activemq::core::RedeliveryPolicy, 3126
  - activemq::transport::failover::FailoverTransport, 1845
- setUseParentHandlers
  - decaf::util::logging::Logger, 2356
- setUserID
  - activemq::commands::Message, 2490
- setUserInfo
  - decaf::internal::net::URIType, 3891
- setUserName
  - activemq::commands::ConnectionInfo, 1329
- setUsername
  - activemq::core::ActiveMQConnection, 263
  - activemq::core::ActiveMQConnectionFactory, 274
- setValid
  - decaf::internal::net::URIType, 3891
- setValue
  - activemq::commands::BrokerId, 831
  - activemq::commands::ConnectionId, 1300
  - activemq::commands::ConsumerId, 1401
  - activemq::commands::LocalTransactionId, 2310
  - activemq::commands::MessageId, 2627
  - activemq::commands::ProducerId, 3018
  - activemq::commands::SessionId, 3323
  - activemq::util::PrimitiveValueNode, 2974
  - decaf::util::Map::Entry, 1789
- setVersion
  - activemq::commands::WireFormatInfo, 3021
  - activemq::wireformat::openwire::OpenWireFormat, 2847
  - activemq::wireformat::stomp::StompWireFormat, 3588
  - activemq::wireformat::WireFormat, 3910
- setWireFormat
  - activemq::core::policies::DefaultRedeliveryPolicy, 1648
  - decaf::internal::net::ssl::openssl::OpenSSLParameters, 2797

- decaf::internal::net::ssl::openssl::OpenSSLServerSocket, 3814
- 2802
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 1196
- 2819
- decaf::net::ssl::SSLParameters, 3498
- 1639
- decaf::net::ssl::SSLServerSocket, 3504
- decaf::net::ssl::SSLSocket, 3514
- ShutdownInfo
- setWasPrepared
- activemq::commands::JournalTransactionShutdownInfoMarshaller
- 2201
- setWindowSize
- 3425
- activemq::commands::ProducerInfo, 3046
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller
- setWireFormat
- 3421
- activemq::transport::failover::FailoverTransport, 1845
- 3433
- activemq::transport::IOTransport, 2111
- activemq::transport::mock::MockTransport, 2733
- 3437
- activemq::transport::Transport, 3824
- 3429
- activemq::transport::TransportFilter, 3834
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller
- activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller
- setWriteCheckTime
- 3417
- activemq::transport::inactivity::InactivityMonitor
- 1967
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2820
- SEVERE
- 2820
- decaf::util::logging::Level, 2295
- decaf::internal::net::tcp::TcpSocket, 3689
- severe
- decaf::net::Socket, 3462
- decaf::util::logging::Logger, 2356
- decaf::net::SocketImpl, 3479
- Short
- shutdownLibrary
- decaf::lang::Short, 3382
- activemq::library::ActiveMQCPP, 293
- SHORT\_TYPE
- shutdownNetworking
- activemq::util::PrimitiveValueNode, 2963
- decaf::internal::net::Network, 2746
- ShortArrayBuffer
- shutdownOutput
- decaf::internal::nio::ShortArrayBuffer, 3393–3395
- decaf::internal::net::ssl::openssl::OpenSSLSocket, 2820
- ShortBuffer
- decaf::internal::net::tcp::TcpSocket, 3690
- decaf::nio::ShortBuffer, 3403
- decaf::net::Socket, 3462
- shortValue
- decaf::net::SocketImpl, 3480
- activemq::util::PrimitiveValueNode::PrimitiveValueRuntime
- 2958
- decaf::lang::Byte, 926
- signal
- decaf::lang::Character, 1076
- decaf::util::concurrent::locks::Condition, 1226
- decaf::lang::Double, 1759
- signalAll
- decaf::lang::Float, 1873
- decaf::util::concurrent::locks::Condition, 1226
- decaf::lang::Integer, 2049
- SignatureException
- decaf::lang::Long, 2388
- decaf::security::SignatureException, 3441, 3442
- decaf::lang::Number, 2788
- shutdown
- 3442
- activemq::state::ConnectionState, 1361
- activemq::state::SessionState, 3379
- decaf::lang::Integer, 2050
- decaf::lang::Short, 3388
- decaf::lang::Integer, 2050

- decaf::lang::Long, 2388
- decaf::lang::Math, 2466, 2467
- SimpleFormatter
  - decaf::util::logging::SimpleFormatter, 3443
- SimpleLogger
  - decaf::util::logging::SimpleLogger, 3444
- SIZE
  - decaf::lang::Byte, 928
  - decaf::lang::Character, 1077
  - decaf::lang::Double, 1762
  - decaf::lang::Float, 1876
  - decaf::lang::Integer, 2054
  - decaf::lang::Long, 2392
  - decaf::lang::Short, 3389
- size
  - activemq::commands::ProducerAck, 2987
  - activemq::core::MessageDispatchChannel, 2564
  - activemq::wireformat::openwire::utils::HexTable, 1948
  - decaf::internal::util::TimerTaskHeap, 3747
  - decaf::io::ByteArrayOutputStream, 994
  - decaf::io::DataOutputStream, 1549
  - decaf::util::Collection, 1164
  - decaf::util::concurrent::ConcurrentStlMap, 1217
  - decaf::util::concurrent::SynchronousQueue, 3669
  - decaf::util::Map, 2430
  - decaf::util::PriorityQueue, 2983
  - decaf::util::Properties, 3080
  - decaf::util::StlList, 3542
  - decaf::util::StlMap, 3553
  - decaf::util::StlQueue, 3562
  - decaf::util::StlSet, 3571
  - gz\_state, 1940
- skip
  - decaf::internal::net::ssl::openssl::OpenSSL::SSLInputStream, 2834
  - decaf::internal::net::tcp::TcpSocketInputStream, 3693
  - decaf::io::BlockingByteArrayInputStream, 803
  - decaf::io::BufferedInputStream, 898
  - decaf::io::ByteArrayInputStream, 991
  - decaf::io::FilterInputStream, 1859
  - decaf::io::InputStream, 2010
  - decaf::io::PushbackInputStream, 3091
  - decaf::io::Reader, 3114
  - decaf::util::zip::CheckedInputStream, 1130
  - decaf::util::zip::InflaterInputStream, 2000
  - gz\_state, 1940
  - skipBytes
    - decaf::io::DataInput, 1531
    - decaf::io::DataInputStream, 1540
  - slaveBroker
    - activemq::commands::BrokerInfo, 862
  - sleep
    - decaf::lang::Thread, 3714
    - decaf::util::concurrent::TimeUnit, 3752
  - SLEEPING
    - decaf::lang::Thread, 3710
  - slice
    - decaf::internal::nio::ByteBuffer, 984
    - decaf::internal::nio::CharArrayBuffer, 1088
    - decaf::internal::nio::DoubleArrayBuffer, 1773
    - decaf::internal::nio::FloatArrayBuffer, 1887
    - decaf::internal::nio::IntArrayBuffer, 2025
    - decaf::internal::nio::LongArrayBuffer, 2402
    - decaf::internal::nio::ShortArrayBuffer, 3400
    - decaf::nio::ByteBuffer, 1021
    - decaf::nio::CharBuffer, 1105
    - decaf::nio::DoubleBuffer, 1784
    - decaf::nio::FloatBuffer, 1898
    - decaf::nio::IntBuffer, 2037
    - decaf::nio::LongBuffer, 2414
    - decaf::nio::ShortBuffer, 3411
  - Socket
    - decaf::net::Socket, 3449–3451
  - SOCKET\_OPTION\_BINDADDR
    - decaf::net::SocketOptions, 3483
  - SOCKET\_OPTION\_BROADCAST
    - decaf::net::SocketOptions, 3484
  - SOCKET\_OPTION\_IP\_MULTICAST\_IF
    - decaf::net::SocketOptions, 3484
  - SOCKET\_OPTION\_IP\_MULTICAST\_IF2
    - decaf::net::SocketOptions, 3484
  - SOCKET\_OPTION\_IP\_MULTICAST\_LOOP
    - decaf::net::SocketOptions, 3484
  - SOCKET\_OPTION\_IP\_TOS
    - decaf::net::SocketOptions, 3484
  - SOCKET\_OPTION\_KEEPALIVE
    - decaf::net::SocketOptions, 3485
  - SOCKET\_OPTION\_LINGER
    - decaf::net::SocketOptions, 3485
  - SOCKET\_OPTION\_OOBLINE
    - decaf::net::SocketOptions, 3485
  - SOCKET\_OPTION\_RCVBUF
    - decaf::net::SocketOptions, 3485

decaf::net::SocketOptions, 3485	src/main/activemq/commands/ActiveMQBytesMessage.h,
SOCKET_OPTION_REUSEADDR	4002
decaf::net::SocketOptions, 3486	src/main/activemq/commands/ActiveMQDestination.h,
SOCKET_OPTION_SNDBUF	4003
decaf::net::SocketOptions, 3486	src/main/activemq/commands/ActiveMQMapMessage.h,
SOCKET_OPTION_TCP_NODELAY	4004
decaf::net::SocketOptions, 3486	src/main/activemq/commands/ActiveMQMessage.h,
SOCKET_OPTION_TIMEOUT	4004
decaf::net::SocketOptions, 3486	src/main/activemq/commands/ActiveMQMessageTemplate.h,
SocketException	4005
decaf::net::SocketException, 3465, 3466	src/main/activemq/commands/ActiveMQObjectMessage.h,
SocketFactory	4006
decaf::net::SocketFactory, 3468	src/main/activemq/commands/ActiveMQQueue.h,
SocketFileDescriptor	4006
decaf::internal::net::SocketFileDescriptor, 3472	src/main/activemq/commands/ActiveMQStreamMessage.h,
SocketImpl	4007
decaf::net::SocketImpl, 3474	src/main/activemq/commands/ActiveMQTempDestination.h,
SocketTimeoutException	4008
decaf::net::SocketTimeoutException, 3487, 3488	src/main/activemq/commands/ActiveMQTempQueue.h,
sqrt	4009
decaf::lang::Math, 2468	src/main/activemq/commands/ActiveMQTextMessage.h,
src/main/activemq/cmsutil/CachedConsumer.h,	4009
3995	src/main/activemq/commands/ActiveMQTopic.h,
src/main/activemq/cmsutil/CachedProducer.h,	4010
3995	src/main/activemq/commands/BaseCommand.h,
src/main/activemq/cmsutil/CmsAccessor.h,	4010
3996	src/main/activemq/commands/BaseDataStructure.h,
src/main/activemq/cmsutil/CmsDestinationAccessor.h,	4011
3996	src/main/activemq/commands/BooleanExpression.h,
src/main/activemq/cmsutil/CmsTemplate.h,	4011
3997	src/main/activemq/commands/BrokerError.h,
src/main/activemq/cmsutil/DestinationResolver.h,	4012
3997	src/main/activemq/commands/BrokerId.h, 4012
src/main/activemq/cmsutil/DynamicDestinationResolver.h,	4013
3998	src/main/activemq/commands/BrokerInfo.h,
src/main/activemq/cmsutil/MessageCreator.h,	4013
3998	src/main/activemq/commands/Command.h,
src/main/activemq/cmsutil/PooledSession.h,	4014
3999	src/main/activemq/commands/ConnectionControl.h,
src/main/activemq/cmsutil/ProducerCallback.h,	4014
3999	src/main/activemq/commands/ConnectionError.h,
src/main/activemq/cmsutil/ResourceLifecycleManager.h,	4015
4000	src/main/activemq/commands/ConnectionId.h,
src/main/activemq/cmsutil/SessionCallback.h,	4015
4001	src/main/activemq/commands/ConnectionInfo.h,
src/main/activemq/cmsutil/SessionPool.h,	4016
src/main/activemq/commands/ActiveMQBlobMessage.h,	4016
4002	src/main/activemq/commands/ConsumerId.h,

4016	4031
src/main/activemq/commands/ConsumerInfo.h,	src/main/activemq/commands/ProducerAck.h,
4017	4031
src/main/activemq/commands/ControlCommand.h,	src/main/activemq/commands/ProducerId.h,
4018	4032
src/main/activemq/commands/DataArrayResponse.h,	src/main/activemq/commands/ProducerInfo.h,
4018	4032
src/main/activemq/commands/DataResponse.h,	src/main/activemq/commands/RemoveInfo.h,
4019	4033
src/main/activemq/commands/DataStructure.h,	src/main/activemq/commands/RemoveSubscriptionInfo.h,
4019	4034
src/main/activemq/commands/DestinationInfo.h,	src/main/activemq/commands/ReplayCommand.h,
4020	4034
src/main/activemq/commands/DiscoveryEvent.h,	src/main/activemq/commands/Response.h,
4020	4035
src/main/activemq/commands/ExceptionResponse.h,	src/main/activemq/commands/SessionId.h,
4021	4035
src/main/activemq/commands/FlushCommand.h,	src/main/activemq/commands/SessionInfo.h,
4021	4036
src/main/activemq/commands/IntegerResponse.h,	src/main/activemq/commands/ShutdownInfo.h,
4022	4036
src/main/activemq/commands/JournalQueueAck.h,	src/main/activemq/commands/SubscriptionInfo.h,
4022	4037
src/main/activemq/commands/JournalTopicAck.h,	src/main/activemq/commands/TransactionId.h,
4023	4037
src/main/activemq/commands/JournalTrace.h,	src/main/activemq/commands/TransactionInfo.h,
4023	4038
src/main/activemq/commands/JournalTransactionId.h,	src/main/activemq/commands/WireFormatInfo.h,
4024	4038
src/main/activemq/commands/KeepAliveInfo.h,	src/main/activemq/commands/XATransactionId.h,
4024	4039
src/main/activemq/commands/LastPartialConsumerId.h,	src/main/activemq/core/ActiveMQAckHandler.h,
4025	4039
src/main/activemq/commands/LocalTransactionId.h,	src/main/activemq/core/ActiveMQConnection.h,
4025	4040
src/main/activemq/commands/Message.h,	src/main/activemq/core/ActiveMQConnectionFactory.h,
4026	4040
src/main/activemq/commands/MessageAck.h,	src/main/activemq/core/ActiveMQConnectionMetaData.h,
4027	4041
src/main/activemq/commands/MessageDispatchInfo.h,	src/main/activemq/core/ActiveMQConstants.h,
4028	4041
src/main/activemq/commands/MessageDispatchInfo.h,	src/main/activemq/core/ActiveMQConsumer.h,
4029	4042
src/main/activemq/commands/MessageId.h,	src/main/activemq/core/ActiveMQProducer.h,
4029	4043
src/main/activemq/commands/MessagePull.h,	src/main/activemq/core/ActiveMQQueueBrowser.h,
4030	4043
src/main/activemq/commands/NetworkBridgeFilter.h,	src/main/activemq/core/ActiveMQSession.h,
4030	4044
src/main/activemq/commands/PartialCommand.h,	src/main/activemq/core/ActiveMQSessionExecutor.h,

4045 src/main/activemq/threads/TaskRunner.h, 4065  
 src/main/activemq/core/ActiveMQTransactionsCommand.h, 4046  
 4046 src/main/activemq/transport/AbstractTransportFactory.h, 4065  
 src/main/activemq/core/DispatchData.h, 4046 src/main/activemq/transport/CompositeTransport.h, 4066  
 src/main/activemq/core/Dispatcher.h, 4047  
 src/main/activemq/core/MessageDispatchChannel.h, 4047 src/main/activemq/transport/correlator/FutureResponse.h, 4067  
 src/main/activemq/core/policies/DefaultPrefetchPolicy.h, 4048 src/main/activemq/transport/correlator/ResponseCorrelator.h, 4067  
 4048 src/main/activemq/transport/DefaultTransportListener.h, 4068  
 src/main/activemq/core/policies/DefaultRedeliveryPolicy.h, 4048  
 4048 src/main/activemq/transport/failover/BackupTransport.h, 4068  
 src/main/activemq/core/PrefetchPolicy.h, 4049 src/main/activemq/transport/failover/BackupTransportPool.h, 4069  
 src/main/activemq/core/RedeliveryPolicy.h, 4049  
 4049 src/main/activemq/transport/failover/CloseTransportsTask.h, 4070  
 src/main/activemq/core/Synchronization.h, 4050  
 4050 src/main/activemq/transport/failover/FailoverTransport.h, 4070  
 src/main/activemq/exceptions/ActiveMQException.h, 4050  
 4050 src/main/activemq/transport/failover/FailoverTransportFactory.h, 4071  
 src/main/activemq/exceptions/BrokerException.h, 4051  
 4051 src/main/activemq/transport/failover/FailoverTransportListener.h, 4071  
 src/main/activemq/exceptions/ExceptionDefines.h, 4051  
 4051 src/main/activemq/transport/failover/URIPool.h, 4072  
 src/main/activemq/io/LoggingInputStream.h, 4055  
 4055 src/main/activemq/transport/inactivity/InactivityMonitor.h, 4073  
 src/main/activemq/io/LoggingOutputStream.h, 4056  
 4056 src/main/activemq/transport/inactivity/ReadChecker.h, 4073  
 src/main/activemq/library/ActiveMQCPP.h, 4056  
 4056 src/main/activemq/transport/inactivity/WriteChecker.h, 4074  
 src/main/activemq/state/CommandVisitor.h, 4057  
 4057 src/main/activemq/transport/IOTransport.h, 4074  
 src/main/activemq/state/CommandVisitorAdapter.h, 4057  
 4057 src/main/activemq/transport/logging/LoggingTransport.h, 4075  
 src/main/activemq/state/ConnectionState.h, 4059  
 4059 src/main/activemq/transport/mock/InternalCommandListener.h, 4076  
 src/main/activemq/state/ConnectionStateTracker.h, 4059  
 4059 src/main/activemq/transport/mock/MockTransport.h, 4076  
 src/main/activemq/state/ConsumerState.h, 4060  
 4060 src/main/activemq/transport/mock/MockTransportFactory.h, 4077  
 src/main/activemq/state/ProducerState.h, 4061  
 4061 src/main/activemq/transport/mock/ResponseBuilder.h, 4078  
 src/main/activemq/state/SessionState.h, 4061  
 4061 src/main/activemq/transport/tcp/SslTransport.h, 4078  
 src/main/activemq/state/Tracked.h, 4062  
 4062 src/main/activemq/transport/tcp/SslTransportFactory.h, 4079  
 src/main/activemq/state/TransactionState.h, 4062  
 4062 src/main/activemq/transport/tcp/TcpTransport.h, 4079  
 src/main/activemq/threads/CompositeTask.h, 4063  
 4063 src/main/activemq/transport/tcp/TcpTransportFactory.h, 4079  
 src/main/activemq/threads/CompositeTaskRunner.h, 4063  
 4063 src/main/activemq/transport/tcp/TcpTransport.h, 4079  
 src/main/activemq/threads/DedicatedTaskRunner.h, 4064  
 4064 src/main/activemq/transport/tcp/TcpTransportFactory.h, 4079  
 src/main/activemq/threads/Task.h, 4065



4080	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQQueueMarshaller.h
src/main/activemq/transport/Transport.h, 4081	4121
src/main/activemq/transport/TransportFactory.h, 4081	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQStreamMessageMarshaller.h, 4125
src/main/activemq/transport/TransportFilter.h, 4082	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempDestinationMarshaller.h, 4130
src/main/activemq/transport/TransportListener.h, 4082	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempQueueMarshaller.h, 4134
src/main/activemq/transport/TransportRegistry.h, 4083	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTempTopicMarshaller.h, 4139
src/main/activemq/util/ActiveMQProperties.h, 4083	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTextMessageMarshaller.h, 4143
src/main/activemq/util/CMSExceptionSupport.h, 4084	src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQTopicMarshaller.h, 4147
src/main/activemq/util/CompositeData.h, 4086	src/main/activemq/wireformat/openwire/marshall/v1/BaseCommandMarshaller.h, 4152
src/main/activemq/util/Config.h, 4086	
src/main/activemq/util/IdGenerator.h, 4087	src/main/activemq/wireformat/openwire/marshall/v1/BrokerIdMarshaller.h, 4156
src/main/activemq/util/LongSequenceGenerator.h, 4088	src/main/activemq/wireformat/openwire/marshall/v1/BrokerInfoMarshaller.h, 4160
src/main/activemq/util/MarshallingSupport.h, 4088	src/main/activemq/wireformat/openwire/marshall/v1/ConnectionControlMarshaller.h, 4164
src/main/activemq/util/MemoryUsage.h, 4089	src/main/activemq/wireformat/openwire/marshall/v1/ConnectionErrorMarshaller.h, 4169
src/main/activemq/util/PrimitiveList.h, 4089	
src/main/activemq/util/PrimitiveMap.h, 4090	src/main/activemq/wireformat/openwire/marshall/v1/ConnectionIdMarshaller.h, 4173
src/main/activemq/util/PrimitiveValueConverter.h, 4090	src/main/activemq/wireformat/openwire/marshall/v1/ConnectionInfoMarshaller.h, 4177
src/main/activemq/util/PrimitiveValueNode.h, 4091	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerControlMarshaller.h, 4182
src/main/activemq/util/URISupport.h, 4091	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerIdMarshaller.h, 4186
src/main/activemq/util/Usage.h, 4092	src/main/activemq/wireformat/openwire/marshall/v1/ConsumerInfoMarshaller.h, 4190
src/main/activemq/wireformat/MarshalAware.h, 4092	src/main/activemq/wireformat/openwire/marshall/v1/ControlCommandMarshaller.h, 4195
src/main/activemq/wireformat/openwire/marshall/v1/BaseDataResponseMarshaller.h, 4093	src/main/activemq/wireformat/openwire/marshall/v1/DataArrayResponseMarshaller.h, 4199
src/main/activemq/wireformat/openwire/marshall/v1/DataStreamMarshaller.h, 4093	src/main/activemq/wireformat/openwire/marshall/v1/DataResponseMarshaller.h, 4203
src/main/activemq/wireformat/openwire/marshall/v1/PrimitiveTypesMarshaller.h, 4094	src/main/activemq/wireformat/openwire/marshall/v1/DestinationInfoMarshaller.h, 4208
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQObjectMessageMarshaller.h, 4095	src/main/activemq/wireformat/openwire/marshall/v1/DiscoveryEventMarshaller.h, 4212
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQQueueMessageMarshaller.h, 4099	src/main/activemq/wireformat/openwire/marshall/v1/ExceptionResponseMarshaller.h, 4217
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQDestinationMarshaller.h, 4104	src/main/activemq/wireformat/openwire/marshall/v1/FlushCommandMarshaller.h, 4221
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQObjectMessageMarshaller.h, 4108	
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQQueueMessageMarshaller.h, 4112	src/main/activemq/wireformat/openwire/marshall/v1/IntegerResponseMarshaller.h, 4225
src/main/activemq/wireformat/openwire/marshall/v1/ActiveMQObjectMessageMarshaller.h, 4117	

src/main/activemq/wireformat/openwire/marshall/marshall/v1/ShutdownInfoMarshaller	4230	src/main/activemq/wireformat/openwire/marshall/marshall/v1/ShutdownInfoMarshaller	4336
src/main/activemq/wireformat/openwire/marshall/marshall/v1/SubscriptionInfoMarshaller	4234	src/main/activemq/wireformat/openwire/marshall/marshall/v1/SubscriptionInfoMarshaller	4340
src/main/activemq/wireformat/openwire/marshall/marshall/v1/TransactionIdMarshaller	4238	src/main/activemq/wireformat/openwire/marshall/marshall/v1/TransactionIdMarshaller	4344
src/main/activemq/wireformat/openwire/marshall/marshall/v1/TransactionInfoMarshaller	4243	src/main/activemq/wireformat/openwire/marshall/marshall/v1/TransactionInfoMarshaller	4349
src/main/activemq/wireformat/openwire/marshall/marshall/v1/WireFormatInfoMarshaller	4247	src/main/activemq/wireformat/openwire/marshall/marshall/v1/WireFormatInfoMarshaller	4353
src/main/activemq/wireformat/openwire/marshall/marshall/v1/XATransactionIdMarshaller	4251	src/main/activemq/wireformat/openwire/marshall/marshall/v1/XATransactionIdMarshaller	4357
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQBlobMarshaller	4256	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQBlobMarshaller	4096
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQBytesMarshaller	4260	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQBytesMarshaller	4100
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQDestinationMarshaller	4263	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQDestinationMarshaller	4104
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQMapMarshaller	4267	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQMapMarshaller	4109
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQMessageMarshaller	4272	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQMessageMarshaller	4113
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQObjectMarshaller	4276	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQObjectMarshaller	4117
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQQueueMarshaller	4281	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQQueueMarshaller	4122
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQStreamMarshaller	4285	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQStreamMarshaller	4126
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTempDestinationMarshaller	4289	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTempDestinationMarshaller	4130
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTempQueueMarshaller	4293	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTempQueueMarshaller	4135
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTempTopicMarshaller	4298	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTempTopicMarshaller	4139
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTextMarshaller	4302	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTextMarshaller	4144
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTopicMarshaller	4306	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ActiveMQTopicMarshaller	4148
src/main/activemq/wireformat/openwire/marshall/marshall/v2/BaseCommandMarshaller	4311	src/main/activemq/wireformat/openwire/marshall/marshall/v2/BaseCommandMarshaller	4152
src/main/activemq/wireformat/openwire/marshall/marshall/v2/BrokerIdMarshaller	4315	src/main/activemq/wireformat/openwire/marshall/marshall/v2/BrokerIdMarshaller	4157
src/main/activemq/wireformat/openwire/marshall/marshall/v2/BrokerInfoMarshaller	4319	src/main/activemq/wireformat/openwire/marshall/marshall/v2/BrokerInfoMarshaller	4161
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ConnectionContainerMarshaller	4323	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ConnectionContainerMarshaller	4165
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ConnectionErrorMarshaller	4328	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ConnectionErrorMarshaller	4169
src/main/activemq/wireformat/openwire/marshall/marshall/v2/ConnectionIdMarshaller	4332	src/main/activemq/wireformat/openwire/marshall/marshall/v2/ConnectionIdMarshaller	4174

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatMessageMarshaller	4235
4127	4235
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMarshaller	4240
4131	4240
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4244
4136	4244
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicInfoMarshaller	4249
4140	4249
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4253
4144	4253
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4257
4149	4257
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4261
4153	4261
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4265
4157	4265
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4269
4162	4269
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4273
4166	4273
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4278
4170	4278
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4282
4174	4282
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4286
4179	4286
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4290
4183	4290
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4295
4187	4295
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4299
4192	4299
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4303
4196	4303
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4308
4201	4308
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4312
4205	4312
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4316
4209	4316
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4320
4214	4320
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4325
4218	4325
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4329
4222	4329
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4333
4227	4333
src/main/activemq/wireformat/openwire/marshall/v3/ActiveMQWireFormatTopicMessageMarshaller	4337
4231	4337

src/main/activemq/wireformat/openwire/marshaller/v3/SubscriptionWireFormatOpenWireMarshaller.h	4341	src/main/activemq/wireformat/openwire/marshaller/v4/ConsumerControlMarshaller.h	4184
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4346	src/main/activemq/wireformat/openwire/marshaller/v4/ConsumerIdMarshaller.h	4188
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4350	src/main/activemq/wireformat/openwire/marshaller/v4/ConsumerInfoMarshaller.h	4193
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4354	src/main/activemq/wireformat/openwire/marshaller/v4/ControlCommandMarshaller.h	4197
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4359	src/main/activemq/wireformat/openwire/marshaller/v4/DataArrayResponseMarshaller.h	4201
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4097	src/main/activemq/wireformat/openwire/marshaller/v4/DataResponseMarshaller.h	4206
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4101	src/main/activemq/wireformat/openwire/marshaller/v4/DestinationInfoMarshaller.h	4210
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4106	src/main/activemq/wireformat/openwire/marshaller/v4/DiscoveryEventMarshaller.h	4214
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4110	src/main/activemq/wireformat/openwire/marshaller/v4/ExceptionResponseMarshaller.h	4219
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4114	src/main/activemq/wireformat/openwire/marshaller/v4/FlushCommandMarshaller.h	4223
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4119	src/main/activemq/wireformat/openwire/marshaller/v4/IntegerResponseMarshaller.h	4227
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4123	src/main/activemq/wireformat/openwire/marshaller/v4/JournalQueueAckMarshaller.h	4232
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4128	src/main/activemq/wireformat/openwire/marshaller/v4/JournalTopicAckMarshaller.h	4236
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4132	src/main/activemq/wireformat/openwire/marshaller/v4/JournalTraceMarshaller.h	4241
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4136	src/main/activemq/wireformat/openwire/marshaller/v4/JournalTransactionMarshaller.h	4245
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4141	src/main/activemq/wireformat/openwire/marshaller/v4/KeepAliveInfoMarshaller.h	4249
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4145	src/main/activemq/wireformat/openwire/marshaller/v4/LastPartialCommandMarshaller.h	4254
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4149	src/main/activemq/wireformat/openwire/marshaller/v4/LocalTransactionIdMarshaller.h	4258
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4154	src/main/activemq/wireformat/openwire/marshaller/v4/MarshallerFactory.h	4262
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4158	src/main/activemq/wireformat/openwire/marshaller/v4/MessageAckMarshaller.h	4265
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4162	src/main/activemq/wireformat/openwire/marshaller/v4/MessageDispatchMarshaller.h	4270
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4166	src/main/activemq/wireformat/openwire/marshaller/v4/MessageDispatchNotificationMarshaller.h	4274
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4171	src/main/activemq/wireformat/openwire/marshaller/v4/MessageIdMarshaller.h	4278
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4175	src/main/activemq/wireformat/openwire/marshaller/v4/MessageMarshaller.h	4283
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionWireFormatOpenWireMarshaller.h	4179	src/main/activemq/wireformat/openwire/marshaller/v4/MessagePullMarshaller.h	4287

src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBridgeFilterIdMarshaller	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempD
4291	4133
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempD	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempC
4295	4137
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempT	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempT
4300	4141
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTextM	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTextM
4304	4146
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTopicM	src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTopicM
4308	4150
src/main/activemq/wireformat/openwire/marshall/v5/BaseCommand	src/main/activemq/wireformat/openwire/marshall/v5/BaseCommand
4313	4155
src/main/activemq/wireformat/openwire/marshall/v5/BrokerIdMarsha	src/main/activemq/wireformat/openwire/marshall/v5/BrokerIdMarsha
4317	4159
src/main/activemq/wireformat/openwire/marshall/v5/BrokerInfoMarsh	src/main/activemq/wireformat/openwire/marshall/v5/BrokerInfoMarsh
4321	4163
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionCont	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionCont
4325	4167
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionError	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionError
4330	4171
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionIdMa	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionIdMa
4334	4176
src/main/activemq/wireformat/openwire/marshall/v5/ConnectionInfoM	src/main/activemq/wireformat/openwire/marshall/v5/ConnectionInfoM
4338	4180
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerContr	src/main/activemq/wireformat/openwire/marshall/v5/ConsumerContr
4342	4185
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerIdMar	src/main/activemq/wireformat/openwire/marshall/v5/ConsumerIdMar
4346	4189
src/main/activemq/wireformat/openwire/marshall/v5/ConsumerInfoM	src/main/activemq/wireformat/openwire/marshall/v5/ConsumerInfoM
4351	4193
src/main/activemq/wireformat/openwire/marshall/v5/ControlComm	src/main/activemq/wireformat/openwire/marshall/v5/ControlComm
4355	4198
src/main/activemq/wireformat/openwire/marshall/v5/DataArrayRespo	src/main/activemq/wireformat/openwire/marshall/v5/DataArrayRespo
4360	4202
src/main/activemq/wireformat/openwire/marshall/v5/DataResponseM	src/main/activemq/wireformat/openwire/marshall/v5/DataResponseM
4098	4206
src/main/activemq/wireformat/openwire/marshall/v5/DestinationInfoM	src/main/activemq/wireformat/openwire/marshall/v5/DestinationInfoM
4102	4211
src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEvent	src/main/activemq/wireformat/openwire/marshall/v5/DiscoveryEvent
4106	4215
src/main/activemq/wireformat/openwire/marshall/v5/ExceptionRespo	src/main/activemq/wireformat/openwire/marshall/v5/ExceptionRespo
4111	4219
src/main/activemq/wireformat/openwire/marshall/v5/FlushCommand	src/main/activemq/wireformat/openwire/marshall/v5/FlushCommand
4115	4224
src/main/activemq/wireformat/openwire/marshall/v5/IntegerRespon	src/main/activemq/wireformat/openwire/marshall/v5/IntegerRespon
4120	4228
src/main/activemq/wireformat/openwire/marshall/v5/JournalQueueA	src/main/activemq/wireformat/openwire/marshall/v5/JournalQueueA
4124	4233
src/main/activemq/wireformat/openwire/marshall/v5/JournalTopicAck	src/main/activemq/wireformat/openwire/marshall/v5/JournalTopicAck
4128	4237

src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQBlobMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQBlobMessageMarshaller.h,
4241	4347
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQDestinationMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQDestinationMarshaller.h,
4246	4352
src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQWireFormatInfoMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v5/ActiveMQWireFormatInfoMarshaller.h,
4250	4356
src/main/activemq/wireformat/openwire/marshaller/v5/XATransactionIdMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v5/XATransactionIdMarshaller.h,
4254	4360
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBlobMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBlobMessageMarshaller.h,
4259	4098
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBytesMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQBytesMessageMarshaller.h,
4262	4103
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQDestinationMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQDestinationMarshaller.h,
4266	4107
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMapMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMapMessageMarshaller.h,
4270	4112
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQMessageMarshaller.h,
4275	4116
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQObjectMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQObjectMessageMarshaller.h,
4279	4120
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQQueueMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQQueueMarshaller.h,
4283	4125
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQStreamMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQStreamMessageMarshaller.h,
4287	4129
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempDestinationMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempDestinationMarshaller.h,
4292	4133
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempQueueMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempQueueMarshaller.h,
4296	4138
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempTopicMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTempTopicMarshaller.h,
4301	4142
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTextMessageMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTextMessageMarshaller.h,
4305	4147
src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ActiveMQTopicMarshaller.h,
4309	4151
src/main/activemq/wireformat/openwire/marshaller/v6/BaseCommandMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/BaseCommandMarshaller.h,
4313	4155
src/main/activemq/wireformat/openwire/marshaller/v6/BrokerIdMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/BrokerIdMarshaller.h,
4317	4160
src/main/activemq/wireformat/openwire/marshaller/v6/BrokerInfoMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/BrokerInfoMarshaller.h,
4322	4164
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionControlMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionControlMarshaller.h,
4326	4168
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionErrorMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionErrorMarshaller.h,
4330	4172
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionIdMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionIdMarshaller.h,
4334	4177
src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionInfoMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ConnectionInfoMarshaller.h,
4338	4181
src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerControlMarshaller.h,	src/main/activemq/wireformat/openwire/marshaller/v6/ConsumerControlMarshaller.h,
4343	4185

src/main/activemq/wireformat/openwire/marshall/v6/PartialCommandMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/PartialCommandMarshaller
4190	4297
src/main/activemq/wireformat/openwire/marshall/v6/ProducerAckMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/ProducerAckMarshaller
4194	4301
src/main/activemq/wireformat/openwire/marshall/v6/ProducerIdMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/ProducerIdMarshaller
4198	4306
src/main/activemq/wireformat/openwire/marshall/v6/ProducerInfoMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/ProducerInfoMarshaller
4203	4310
src/main/activemq/wireformat/openwire/marshall/v6/RemoveInfoMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/RemoveInfoMarshaller
4207	4314
src/main/activemq/wireformat/openwire/marshall/v6/RemoveSubscriptionMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/RemoveSubscriptionMarshaller
4211	4318
src/main/activemq/wireformat/openwire/marshall/v6/ReplayCommandMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/ReplayCommandMarshaller
4216	4323
src/main/activemq/wireformat/openwire/marshall/v6/ResponseMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/ResponseMarshaller
4220	4327
src/main/activemq/wireformat/openwire/marshall/v6/SessionIdMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/SessionIdMarshaller
4225	4331
src/main/activemq/wireformat/openwire/marshall/v6/SessionInfoMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/SessionInfoMarshaller
4229	4335
src/main/activemq/wireformat/openwire/marshall/v6/ShutdownInfoMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/ShutdownInfoMarshaller
4233	4339
src/main/activemq/wireformat/openwire/marshall/v6/SubscriptionInfoMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/SubscriptionInfoMarshaller
4238	4344
src/main/activemq/wireformat/openwire/marshall/v6/TransactionIdMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/TransactionIdMarshaller
4242	4348
src/main/activemq/wireformat/openwire/marshall/v6/TransactionInfoMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/TransactionInfoMarshaller
4246	4352
src/main/activemq/wireformat/openwire/marshall/v6/WireFormatInfoMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/WireFormatInfoMarshaller
4251	4357
src/main/activemq/wireformat/openwire/marshall/v6/XATransactionIdMarshaller	src/main/activemq/wireformat/openwire/marshall/v6/XATransactionIdMarshaller
4255	4361
src/main/activemq/wireformat/openwire/marshall/v6/OpenWireFormat.h	src/main/activemq/wireformat/openwire/marshall/v6/OpenWireFormat.h
4259	4362
src/main/activemq/wireformat/openwire/marshall/v6/OpenWireFormatFactory.h	src/main/activemq/wireformat/openwire/marshall/v6/OpenWireFormatFactory.h
4263	4362
src/main/activemq/wireformat/openwire/marshall/v6/OpenWireFormatNegotiator	src/main/activemq/wireformat/openwire/marshall/v6/OpenWireFormatNegotiator
4267	4363
src/main/activemq/wireformat/openwire/marshall/v6/OpenWireResponseBuilder	src/main/activemq/wireformat/openwire/marshall/v6/OpenWireResponseBuilder
4271	4364
src/main/activemq/wireformat/openwire/marshall/v6/BooleanStream.h	src/main/activemq/wireformat/openwire/marshall/v6/BooleanStream.h
4276	4364
src/main/activemq/wireformat/openwire/marshall/v6/HexTable.h	src/main/activemq/wireformat/openwire/marshall/v6/HexTable.h
4280	4365
src/main/activemq/wireformat/openwire/marshall/v6/MessagePropertyInterceptor	src/main/activemq/wireformat/openwire/marshall/v6/MessagePropertyInterceptor
4284	4365
src/main/activemq/wireformat/openwire/marshall/v6/StompCommandConstants.h	src/main/activemq/wireformat/openwire/marshall/v6/StompCommandConstants.h
4288	4366
src/main/activemq/wireformat/openwire/marshall/v6/StompFrame.h	src/main/activemq/wireformat/openwire/marshall/v6/StompFrame.h
4293	4366



src/main/activemq/wireformat/stomp/StompHeader.h, 4367  
 src/main/activemq/wireformat/stomp/StompWireFormat.h, 4368  
 src/main/activemq/wireformat/stomp/StompWireFormatFactory.h, 4368  
 src/main/activemq/wireformat/WireFormat.h, 4369  
 src/main/activemq/wireformat/WireFormatFactory.h, 4370  
 src/main/activemq/wireformat/WireFormatNegotiator.h, 4370  
 src/main/activemq/wireformat/WireFormatRegistry.h, 4371  
 src/main/cms/BytesMessage.h, 4371  
 src/main/cms/Closeable.h, 4372  
 src/main/cms/CMSEException.h, 4373  
 src/main/cms/CMSProperties.h, 4373  
 src/main/cms/CMSSecurityException.h, 4374  
 src/main/cms/Config.h, 4087  
 src/main/cms/Connection.h, 4374  
 src/main/cms/ConnectionFactory.h, 4375  
 src/main/cms/ConnectionMetaData.h, 4375  
 src/main/cms/DeliveryMode.h, 4375  
 src/main/cms/Destination.h, 4376  
 src/main/cms/ExceptionListener.h, 4376  
 src/main/cms/IllegalStateException.h, 4377  
 src/main/cms/InvalidClientIdException.h, 4378  
 src/main/cms/InvalidDestinationException.h, 4378  
 src/main/cms/InvalidSelectorException.h, 4378  
 src/main/cms/MapMessage.h, 4379  
 src/main/cms/Message.h, 4027  
 src/main/cms/MessageConsumer.h, 4379  
 src/main/cms/MessageEnumeration.h, 4380  
 src/main/cms/MessageEOFException.h, 4380  
 src/main/cms/MessageFormatException.h, 4381  
 src/main/cms/MessageListener.h, 4381  
 src/main/cms/MessageNotReadableException.h, 4382  
 src/main/cms/MessageNotWritableException.h, 4382  
 src/main/cms/MessageProducer.h, 4382  
 src/main/cms/ObjectMessage.h, 4383  
 src/main/cms/Queue.h, 4383  
 src/main/cms/QueueBrowser.h, 4384  
 src/main/cms/Session.h, 4385  
 src/main/cms/Startable.h, 4386  
 src/main/cms/Stoppable.h, 4386  
 src/main/cms/StreamMessage.h, 4387  
 src/main/cms/TemporaryQueue.h, 4387  
 src/main/cms/TemporaryTopic.h, 4388  
 src/main/cms/TextMessage.h, 4388  
 src/main/cms/Topic.h, 4388  
 src/main/cms/UnsupportedOperationException.h, 4389  
 src/main/decaf/internal/AprPool.h, 4390  
 src/main/decaf/internal/DecafRuntime.h, 4390  
 src/main/decaf/internal/io/StandardErrorOutputStream.h, 4391  
 src/main/decaf/internal/io/StandardInputStream.h, 4391  
 src/main/decaf/internal/io/StandardOutputStream.h, 4392  
 src/main/decaf/internal/net/DefaultServerSocketFactory.h, 4392  
 src/main/decaf/internal/net/DefaultSocketFactory.h, 4393  
 src/main/decaf/internal/net/Network.h, 4393  
 src/main/decaf/internal/net/SocketFileDescriptor.h, 4394  
 src/main/decaf/internal/net/ssl/DefaultSSLContext.h, 4394  
 src/main/decaf/internal/net/ssl/DefaultSSLServerSocketFactory.h, 4395  
 src/main/decaf/internal/net/ssl/DefaultSSLSocketFactory.h, 4395  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLContextSpi.h, 4396  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLParameters.h, 4396  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocket.h, 4397  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLServerSocketFactory.h, 4397  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocket.h, 4398  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketException.h, 4398  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketFactory.h, 4399  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketInputStream.h, 4399  
 src/main/decaf/internal/net/ssl/openssl/OpenSSLSocketOutputStream.h, 4400  
 src/main/decaf/internal/net/tcp/TcpSocket.h, 4400  
 src/main/decaf/internal/net/tcp/TcpSocketInputStream.h, 4401

src/main/decaf/internal/net/tcp/TcpSocketOutputStream.h,	src/main/decaf/internal/util/HexStringParser.h,
4402	4416
src/main/decaf/internal/net/URIEncoderDecoder.h,	src/main/decaf/internal/util/Resource.h,
4402	4416
src/main/decaf/internal/net/URIHelper.h,	src/main/decaf/internal/util/ResourceLifecycleManager.h,
4403	4000
src/main/decaf/internal/net/URIType.h,	src/main/decaf/internal/util/TimerTaskHeap.h,
4403	4417
src/main/decaf/internal/nio/BufferFactory.h,	src/main/decaf/internal/util/zip/crc32.h,
4404	4417
src/main/decaf/internal/nio/ByteBuffer.h,	src/main/decaf/internal/util/zip/deflate.h,
4404	4417
src/main/decaf/internal/nio/CharArrayBuffer.h,	src/main/decaf/internal/util/zip/gzguts.h,
4405	4421
src/main/decaf/internal/nio/DoubleArrayBuffer.h,	src/main/decaf/internal/util/zip/inffast.h,
4406	4423
src/main/decaf/internal/nio/FloatArrayBuffer.h,	src/main/decaf/internal/util/zip/infixied.h,
4406	4423
src/main/decaf/internal/nio/IntArrayBuffer.h,	src/main/decaf/internal/util/zip/inflate.h,
4407	4423
src/main/decaf/internal/nio/LongArrayBuffer.h,	src/main/decaf/internal/util/zip/inftrees.h,
4408	4425
src/main/decaf/internal/nio/ShortArrayBuffer.h,	src/main/decaf/internal/util/zip/trees.h,
4408	4426
src/main/decaf/internal/security/unix/SecureRandomImpl.h,	src/main/decaf/internal/util/zip/zconf.h,
4409	4428
src/main/decaf/internal/security/windows/SecureRandomImpl.h,	src/main/decaf/internal/util/zip/zlib.h,
4409	4430
src/main/decaf/internal/util/ByteArrayAdapter.h,	src/main/decaf/internal/util/zip/zutil.h,
4410	4437
src/main/decaf/internal/util/concurrent/ConditionImpl.h,	src/main/decaf/io/BlockingByteArrayInputStream.h,
4411	4440
src/main/decaf/internal/util/concurrent/MutexImpl.h,	src/main/decaf/io/BufferedInputStream.h,
4411	4441
src/main/decaf/internal/util/concurrent/SynchronizableImpl.h,	src/main/decaf/io/BufferedOutputStream.h,
4412	4441
src/main/decaf/internal/util/concurrent/Transfer.h,	src/main/decaf/io/ByteArrayInputStream.h,
4412	4442
src/main/decaf/internal/util/concurrent/TransferQueue.h,	src/main/decaf/io/ByteArrayOutputStream.h,
4413	4442
src/main/decaf/internal/util/concurrent/TransferStack.h,	src/main/decaf/io/Closeable.h,
4413	4372
src/main/decaf/internal/util/concurrent/unix/ConditionImpl.h,	src/main/decaf/io/DataInput.h,
4414	4443
src/main/decaf/internal/util/concurrent/unix/MutexImpl.h,	src/main/decaf/io/DataInputStream.h,
4415	4443
src/main/decaf/internal/util/concurrent/windows/ConditionImpl.h,	src/main/decaf/io/DataOutput.h,
4414	4444
src/main/decaf/internal/util/concurrent/windows/MutexImpl.h,	src/main/decaf/io/DataOutputStream.h,
4415	4444
src/main/decaf/internal/util/GenericResources.h,	src/main/decaf/io/EOFException.h,
4415	4445
	src/main/decaf/io/FileDescriptor.h,
	4445
	src/main/decaf/io/FilterInputStream.h,
	4446
	src/main/decaf/io/FilterOutputStream.h,
	4446
	src/main/decaf/io/Flushable.h,
	4447
	src/main/decaf/io/InputStream.h,
	4447
	src/main/decaf/io/InputStreamReader.h,
	4448
	src/main/decaf/io/InterruptedIOException.h,
	4448
	src/main/decaf/io/IOException.h,
	4449
	src/main/decaf/io/OutputStream.h,
	4449
	src/main/decaf/io/OutputStreamWriter.h,
	4450
	src/main/decaf/io/PushbackInputStream.h,
	4450
	src/main/decaf/io/Reader.h,
	4450
	src/main/decaf/io/UnsupportedEncodingException.h,
	4451

src/main/decaf/io/UTFDataFormatException.h, 4451  
 src/main/decaf/io/Writer.h, 4452  
 src/main/decaf/lang/Appendable.h, 4452  
 src/main/decaf/lang/ArrayPointer.h, 4453  
 src/main/decaf/lang/Boolean.h, 4454  
 src/main/decaf/lang/Byte.h, 4454  
 src/main/decaf/lang/Character.h, 4455  
 src/main/decaf/lang/CharSequence.h, 4455  
 src/main/decaf/lang/Comparable.h, 4456  
 src/main/decaf/lang/Double.h, 4456  
 src/main/decaf/lang/Exception.h, 4457  
 src/main/decaf/lang/exceptions/ClassCastException.h, 4457  
 src/main/decaf/lang/exceptions/ExceptionDefines.h, 4053  
 src/main/decaf/lang/exceptions/IllegalArgumentException.h, 4458  
 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h, 4458  
 src/main/decaf/lang/exceptions/IllegalStateException.h, 4377  
 src/main/decaf/lang/exceptions/IllegalThreadStateException.h, 4459  
 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h, 4459  
 src/main/decaf/lang/exceptions/InterruptedException.h, 4459  
 src/main/decaf/lang/exceptions/InvalidStateException.h, 4460  
 src/main/decaf/lang/exceptions/NoSuchElementException.h, 4460  
 src/main/decaf/lang/exceptions/NullPointerException.h, 4461  
 src/main/decaf/lang/exceptions/NumberFormatException.h, 4461  
 src/main/decaf/lang/exceptions/RuntimeIOException.h, 4462  
 src/main/decaf/lang/exceptions/UnsupportedOperationException.h, 4389  
 src/main/decaf/lang/Float.h, 4462  
 src/main/decaf/lang/Integer.h, 4462  
 src/main/decaf/lang/Iterable.h, 4463  
 src/main/decaf/lang/Long.h, 4463  
 src/main/decaf/lang/Math.h, 4464  
 src/main/decaf/lang/Number.h, 4464  
 src/main/decaf/lang/Pointer.h, 4465  
 src/main/decaf/lang/Readable.h, 4466  
 src/main/decaf/lang/Runnable.h, 4466  
 src/main/decaf/lang/Runtime.h, 4467  
 src/main/decaf/lang/Short.h, 4467  
 src/main/decaf/lang/String.h, 4468  
 src/main/decaf/lang/System.h, 4468  
 src/main/decaf/lang/Thread.h, 4469  
 src/main/decaf/lang/ThreadGroup.h, 4469  
 src/main/decaf/lang/Throwable.h, 4470  
 src/main/decaf/net/BindException.h, 4470  
 src/main/decaf/net/ConnectException.h, 4471  
 src/main/decaf/net/HttpRetryException.h, 4471  
 src/main/decaf/net/Inet4Address.h, 4471  
 src/main/decaf/net/Inet6Address.h, 4472  
 src/main/decaf/net/InetAddress.h, 4472  
 src/main/decaf/net/InetSocketAddress.h, 4473  
 src/main/decaf/net/MalformedURLException.h, 4473  
 src/main/decaf/net/NoRouteToHostException.h, 4474  
 src/main/decaf/net/PortUnreachableException.h, 4474  
 src/main/decaf/net/ProtocolException.h, 4474  
 src/main/decaf/net/ServerSocket.h, 4475  
 src/main/decaf/net/ServerSocketFactory.h, 4475  
 src/main/decaf/net/Socket.h, 4476  
 src/main/decaf/net/SocketAddress.h, 4477  
 src/main/decaf/net/SocketError.h, 4477  
 src/main/decaf/net/SocketException.h, 4477  
 src/main/decaf/net/SocketFactory.h, 4478  
 src/main/decaf/net/SocketImpl.h, 4478  
 src/main/decaf/net/SocketImplFactory.h, 4479  
 src/main/decaf/net/SocketOptions.h, 4479  
 src/main/decaf/net/SocketTimeoutException.h, 4480  
 src/main/decaf/net/ssl/SSLContext.h, 4480  
 src/main/decaf/net/ssl/SSLContextSpi.h, 4480  
 src/main/decaf/net/ssl/SSLParameters.h, 4481  
 src/main/decaf/net/ssl/SSLServerSocket.h, 4481  
 src/main/decaf/net/ssl/SSLServerSocketFactory.h, 4482  
 src/main/decaf/net/ssl/SSLSocket.h, 4482  
 src/main/decaf/net/ssl/SSLSocketFactory.h, 4483  
 src/main/decaf/net/UnknownHostException.h, 4483  
 src/main/decaf/net/UnknownServiceException.h, 4484  
 src/main/decaf/net/URI.h, 4484  
 src/main/decaf/net/URISyntaxException.h, 4485

src/main/decaf/net/URL.h, 4485	src/main/decaf/security/SecureRandomSpi.h, 4501
src/main/decaf/net/URLDecoder.h, 4486	
src/main/decaf/net/URLEncoder.h, 4486	src/main/decaf/security/SignatureException.h, 4501
src/main/decaf/nio/Buffer.h, 4486	
src/main/decaf/nio/BufferOverflowException.h, 4487	src/main/decaf/util/AbstractCollection.h, 4501
	src/main/decaf/util/AbstractList.h, 4502
src/main/decaf/nio/BufferUnderflowException.h, 4487	src/main/decaf/util/AbstractMap.h, 4503
	src/main/decaf/util/AbstractQueue.h, 4503
src/main/decaf/nio/ByteBuffer.h, 4488	src/main/decaf/util/AbstractSequentialList.h, 4504
src/main/decaf/nio/CharBuffer.h, 4488	
src/main/decaf/nio/DoubleBuffer.h, 4489	src/main/decaf/util/AbstractSet.h, 4505
src/main/decaf/nio/FloatBuffer.h, 4489	src/main/decaf/util/Collection.h, 4505
src/main/decaf/nio/IntBuffer.h, 4490	src/main/decaf/util/Comparator.h, 4506
src/main/decaf/nio/InvalidMarkException.h, 4490	src/main/decaf/util/comparators/Less.h, 4506
	src/main/decaf/util/concurrent/atomic/AtomicBoolean.h, 4507
src/main/decaf/nio/LongBuffer.h, 4491	
src/main/decaf/nio/ReadOnlyBufferException.h, 4491	src/main/decaf/util/concurrent/atomic/AtomicInteger.h, 4507
src/main/decaf/nio/ShortBuffer.h, 4492	src/main/decaf/util/concurrent/atomic/AtomicReferenceCounter.h, 4508
src/main/decaf/security/auth/x500/X500Principal.h, 4492	src/main/decaf/util/concurrent/atomic/AtomicReference.h, 4508
src/main/decaf/security/cert/Certificate.h, 4493	
src/main/decaf/security/cert/CertificateEncodingException.h, 4494	src/main/decaf/util/concurrent/BlockingQueue.h, 4509
src/main/decaf/security/cert/CertificateException.h, 4494	src/main/decaf/util/concurrent/BrokenBarrierException.h, 4509
src/main/decaf/security/cert/CertificateExpiredException.h, 4494	src/main/decaf/util/concurrent/Callable.h, 4510
	src/main/decaf/util/concurrent/CancellationException.h, 4510
src/main/decaf/security/cert/CertificateNotYetValidException.h, 4495	
	src/main/decaf/util/concurrent/Concurrent.h, 4511
src/main/decaf/security/cert/CertificateParsingException.h, 4495	
	src/main/decaf/util/concurrent/ConcurrentMap.h, 4512
src/main/decaf/security/cert/X509Certificate.h, 4496	src/main/decaf/util/concurrent/ConcurrentStlMap.h, 4512
src/main/decaf/security/GeneralSecurityException.h, 4496	src/main/decaf/util/concurrent/CountDownLatch.h, 4513
src/main/decaf/security/InvalidKeyException.h, 4497	src/main/decaf/util/concurrent/Delayed.h, 4513
src/main/decaf/security/Key.h, 4497	src/main/decaf/util/concurrent/ExecutionException.h, 4514
src/main/decaf/security/KeyException.h, 4498	
src/main/decaf/security/KeyManagementException.h, 4498	src/main/decaf/util/concurrent/Executor.h, 4514
	src/main/decaf/util/concurrent/ExecutorService.h, 4515
src/main/decaf/security/NoSuchAlgorithmException.h, 4498	
	src/main/decaf/util/concurrent/Future.h, 4515
src/main/decaf/security/NoSuchProviderException.h, 4499	src/main/decaf/util/concurrent/Lock.h, 4516
src/main/decaf/security/Principal.h, 4499	src/main/decaf/util/concurrent/locks/Condition.h, 4517
src/main/decaf/security/PublicKey.h, 4500	src/main/decaf/util/concurrent/locks/Lock.h, 4516
src/main/decaf/security/SecureRandom.h, 4500	

src/main/decaf/util/concurrent/locks/LockSupport.h, 4517  
 src/main/decaf/util/concurrent/locks/ReadWriteLock.h, 4518  
 src/main/decaf/util/concurrent/locks/ReentrantLock.h, 4518  
 src/main/decaf/util/concurrent/Mutex.h, 4519  
 src/main/decaf/util/concurrent/PooledThreadPool.h, 4519  
 src/main/decaf/util/concurrent/PooledThreadPoolSystem.h, 4520  
 src/main/decaf/util/concurrent/RejectedExecutionException.h, 4520  
 src/main/decaf/util/concurrent/RejectedExecutionHandler.h, 4521  
 src/main/decaf/util/concurrent/Semaphore.h, 4521  
 src/main/decaf/util/concurrent/Synchronizable.h, 4522  
 src/main/decaf/util/concurrent/SynchronousQueue.h, 4522  
 src/main/decaf/util/concurrent/TaskListener.h, 4523  
 src/main/decaf/util/concurrent/ThreadFactory.h, 4523  
 src/main/decaf/util/concurrent/ThreadPool.h, 4524  
 src/main/decaf/util/concurrent/TimeoutException.h, 4524  
 src/main/decaf/util/concurrent/TimeUnit.h, 4525  
 src/main/decaf/util/Config.h, 4087  
 src/main/decaf/util/Date.h, 4526  
 src/main/decaf/util/Iterator.h, 4526  
 src/main/decaf/util/List.h, 4526  
 src/main/decaf/util/ListIterator.h, 4527  
 src/main/decaf/util/logging/ConsoleHandler.h, 4528  
 src/main/decaf/util/logging/ErrorHandler.h, 4528  
 src/main/decaf/util/logging/Filter.h, 4529  
 src/main/decaf/util/logging/Formatter.h, 4529  
 src/main/decaf/util/logging/Handler.h, 4529  
 src/main/decaf/util/logging/Level.h, 4530  
 src/main/decaf/util/logging/Logger.h, 4531  
 src/main/decaf/util/logging/LoggerCommon.h, 4531  
 src/main/decaf/util/logging/LoggerDefines.h, 4532  
 src/main/decaf/util/logging/LoggerHierarchy.h, 4533  
 src/main/decaf/util/logging/LogManager.h, 4533  
 src/main/decaf/util/logging/LogRecord.h, 4534  
 src/main/decaf/util/logging/LogWriter.h, 4535  
 src/main/decaf/util/logging/MarkBlockLogger.h, 4535  
 src/main/decaf/util/logging/PropertiesChangeListener.h, 4536  
 src/main/decaf/util/logging/SimpleFormatter.h, 4536  
 src/main/decaf/util/logging/SimpleLogger.h, 4537  
 src/main/decaf/util/logging/StreamHandler.h, 4537  
 src/main/decaf/util/logging/XMLFormatter.h, 4538  
 src/main/decaf/util/Map.h, 4538  
 src/main/decaf/util/PriorityQueue.h, 4539  
 src/main/decaf/util/Properties.h, 4539  
 src/main/decaf/util/Queue.h, 4384  
 src/main/decaf/util/Random.h, 4540  
 src/main/decaf/util/Set.h, 4541  
 src/main/decaf/util/StlList.h, 4541  
 src/main/decaf/util/StlMap.h, 4542  
 src/main/decaf/util/StlQueue.h, 4542  
 src/main/decaf/util/StlSet.h, 4543  
 src/main/decaf/util/StringTokenizer.h, 4544  
 src/main/decaf/util/Timer.h, 4544  
 src/main/decaf/util/TimerTask.h, 4545  
 src/main/decaf/util/UUID.h, 4545  
 src/main/decaf/util/zip/Adler32.h, 4546  
 src/main/decaf/util/zip/CheckedInputStream.h, 4546  
 src/main/decaf/util/zip/CheckedOutputStream.h, 4547  
 src/main/decaf/util/zip/Checksum.h, 4547  
 src/main/decaf/util/zip/CRC32.h, 4548  
 src/main/decaf/util/zip/DataFormatException.h, 4548  
 src/main/decaf/util/zip/Deflater.h, 4549  
 src/main/decaf/util/zip/DeflaterOutputStream.h, 4549  
 src/main/decaf/util/zip/Inflater.h, 4550  
 src/main/decaf/util/zip/InflaterInputStream.h, 4550  
 src/main/decaf/util/zip/ZipException.h, 4551  
 SSLContext  
 decaf::net::ssl::SSLContext, 3490  
 SSLParameters  
 decaf::net::ssl::SSLParameters, 3496  
 SSLServerSocket

decaf::net::ssl::SSLServerSocket, 3499,  
 3500  
 SSLServerSocketFactory  
 decaf::net::ssl::SSLServerSocketFactory, 3505  
 SSLSocket  
 decaf::net::ssl::SSLSocket, 3508, 3509  
 SSLSocketFactory  
 decaf::net::ssl::SSLSocketFactory, 3516  
 SslTransport  
 activemq::transport::tcp::SslTransport,  
 3519  
 stackTrace  
 decaf::lang::Exception, 1800  
 StandardErrorOutputStream  
 decaf::internal::io::StandardErrorOutputStream,  
 3522  
 StandardInputStream  
 decaf::internal::io::StandardInputStream,  
 3524  
 StandardOutputStream  
 decaf::internal::io::StandardOutputStream,  
 3526  
 start  
 activemq::commands::ConsumerControl,  
 1373  
 activemq::core::ActiveMQConnection,  
 263  
 activemq::core::ActiveMQConsumer, 291  
 activemq::core::ActiveMQSession, 502  
 activemq::core::ActiveMQSessionExecutor,  
 506  
 activemq::core::MessageDispatchChannel,  
 2564  
 activemq::transport::correlator::ResponseCorrelator,  
 3236  
 activemq::transport::failover::FailoverTransport,  
 1846  
 activemq::transport::IOTransport, 2112  
 activemq::transport::mock::MockTransport,  
 2733  
 activemq::transport::Transport, 3825  
 activemq::transport::TransportFilter, 3834  
 activemq::wireformat::openwire::OpenWireFormatNegotiator,  
 2854  
 cms::Startable, 3527  
 decaf::lang::Thread, 3715  
 gz\_state, 1941  
 startHandshake  
 decaf::internal::net::ssl::openssl::OpenSSLSocket,  
 2820  
 decaf::net::ssl::SSLSocket, 3514  
 stat\_desc  
 tree\_desc\_s, 3840  
 State  
 decaf::lang::Thread, 3709  
 state  
 z\_stream\_s, 3991  
 static\_dtree  
 trees.h, 4428  
 static\_len  
 internal\_state, 2084  
 static\_ltree  
 trees.h, 4428  
 static\_tree\_desc  
 deflate.h, 4421  
 STATIC\_TREES  
 zutil.h, 4439  
 staticCast  
 decaf::lang::Pointer, 2902  
 StaticInitializer  
 activemq::core::ActiveMQConstants::StaticInitializer,  
 3528  
 status  
 internal\_state, 2084  
 std, 145  
 std::binary\_function, 797  
 std::less< decaf::lang::ArrayPointer< T >  
 >, 2289  
 operator(), 2289  
 std::less< decaf::lang::Pointer< T > >, 2289  
 operator(), 2290  
 StIList  
 decaf::util::StIList, 3534  
 StIMap  
 decaf::util::StIMap, 3547  
 StQueue  
 decaf::util::StIQueue, 3558  
 StISet  
 decaf::util::StISet, 3567  
 StompFrame  
 activemq::wireformat::stomp::StompFrame,  
 3578  
 StompHelper  
 activemq::wireformat::stomp::StompHelper,  
 3582  
 StompWireFormat  
 activemq::wireformat::stomp::StompWireFormat,  
 3587

StompWireFormatFactory	activemq::commands::RemoveSubscriptionInfo,
activemq::wireformat::stomp::StompWireFormatFactory,	3169,
3590	activemq::commands::SubscriptionInfo,
stop	3620
activemq::commands::ConsumerControl	SUBSCRIBE
1373	activemq::wireformat::stomp::StompCommandConstants,
activemq::core::ActiveMQConnection,	3576
263	subscribedDestination
activemq::core::ActiveMQConsumer, 291	activemq::commands::SubscriptionInfo,
activemq::core::ActiveMQSession, 502	3620
activemq::core::ActiveMQSessionExecutor	SubscriptionInfo
506	activemq::commands::SubscriptionInfo,
activemq::core::MessageDispatchChannel,	3617
2564	SubscriptionInfoMarshaller
activemq::transport::failover::FailoverTransport	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller,
1846	3625
activemq::transport::IOTransport, 2112	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller,
activemq::transport::mock::MockTransport,	3641
2733	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller,
activemq::transport::Transport, 3825	3621
activemq::transport::TransportFilter, 3835	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,
cms::Stoppable, 3591	3633
decaf::util::concurrent::PooledThread,	activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,
2920	3629
store	activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,
decaf::util::Properties, 3080, 3081	3637
STORED	subscriptionName
inflate.h, 4424	activemq::commands::ConsumerInfo,
STORED_BLOCK	1434
zutil.h, 4439	subscriptionName
strategy	activemq::commands::JournalTopicAck,
gz_state, 1941	2147
internal_state, 2084	subSequence
StreamHandler	decaf::internal::nio::CharArrayBuffer, 1088
decaf::util::logging::StreamHandler, 3593	decaf::lang::CharSequence, 1108
String	decaf::lang::String, 3612
decaf::lang::String, 3611	decaf::nio::CharBuffer, 1105
STRING_TYPE	supportsUrgentData
activemq::util::PrimitiveValueNode, 2964	decaf::net::SocketImpl, 3480
StringTokenizer	suspend
decaf::util::StringTokenizer, 3613	activemq::commands::ConnectionControl,
stringValue	1242
activemq::util::PrimitiveValueNode::PrimitiveValue,	stringValue
2958	decaf::lang::ArrayPointer, 703
strm	decaf::lang::Pointer, 2902
gz_state, 1941	decaf::util::concurrent::atomic::AtomicRefCounter,
internal_state, 2084	715
strstart	SYNC
internal_state, 2084	inflate.h, 4425
subscriptionName	sync

decaf::io::FileDescriptor, 1852	activemq::commands::ActiveMQDestination, 304
SynchronizableImpl	TEMPORARY_QUEUE
decaf::internal::util::concurrent::SynchronizableImpl, 3656	cms::Destination, 1689
synchronized	TEMPORARY_TOPIC
Concurrent.h, 4511	cms::Destination, 1689
SynchronousQueue	TEMPQUEUE_PREFIX
decaf::util::concurrent::SynchronousQueue, 3662	activemq::wireformat::stomp::StompCommandConstants, 3576
syncRequest	TEMPTOPIC_PREFIX
activemq::core::ActiveMQConnection, 263	activemq::wireformat::stomp::StompCommandConstants, 3576
activemq::core::ActiveMQSession, 502	TERMINATED
System	decaf::lang::Thread, 3710
decaf::lang::System, 3672	TEXT
TABLE	activemq::wireformat::stomp::StompCommandConstants, 3576
inflate.h, 4424	text
take	activemq::commands::ActiveMQTextMessage, 635
decaf::util::concurrent::BlockingQueue, 810	gz_header_s, 1939
decaf::util::concurrent::SynchronousQueue, 3669	Thread
takeSession	decaf::lang::Thread, 3710, 3711
activemq::cmsutil::SessionPool, 3377	ThreadGroup
targetConsumerId	decaf::lang::ThreadGroup, 3718
activemq::commands::Message, 2492	ThreadPool
Task	decaf::util::concurrent::ThreadPool, 3720
decaf::util::concurrent::ThreadPool, 3720	Throwable
TcpSocket	decaf::lang::Throwable, 3725
decaf::internal::net::tcp::TcpSocket, 3685	Throwing
TcpSocketInputStream	decaf::util::logging, 144
decaf::internal::net::tcp::TcpSocketInputStream, 3692	decaf::util::logging::Logger, 2356
TcpSocketOutputStream	tightMarshal
decaf::internal::net::tcp::TcpSocketOutputStream, 3695	activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 1606
TcpTransport	activemq::wireformat::openwire::marshal::DataStreamMarshaller, 184
activemq::transport::tcp::TcpTransport, 3697	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessage, 227
TEMP_POSTFIX	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessage, 310
activemq::commands::ActiveMQDestination, 304	activemq::wireformat::openwire::marshal::v1::ActiveMQDestination, 351
TEMP_PREFIX	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessage, 377
activemq::commands::ActiveMQDestination, 304	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessage, 423
TEMP_QUEUE_QUALIFIED_PREFIX	
activemq::commands::ActiveMQDestination, 304	
TEMP_TOPIC_QUALIFIED_PREFIX	



activemq::wireformat::openwire::marshal::v1::ActiveMQQueueFormatMarshaller,	467
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueFormatMarshaller,	2142
activemq::wireformat::openwire::marshal::v1::ActiveMQSimpleMessageMarshaller,	530
activemq::wireformat::openwire::marshal::v1::ActiveMQSimpleMessageMarshaller,	2170
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller,	557
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller,	2193
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller,	585
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller,	2224
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller;	617
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller;	2251
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller,	646
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller,	2285
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,	674
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,	2333
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller,	747
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller,	2545
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	842
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2585
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	873
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2614
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1252
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2650
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1284
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2672
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1315
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2718
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1345
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2772
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1388
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2894
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1416
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3010
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1449
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3041
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1477
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3058
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1511
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3156
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1576
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3172
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1710
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3203
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1743
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3258
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1827
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3347
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	1922
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3362
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	2075
activemq::wireformat::openwire::marshal::v1::ActiveMQWireIdMarshaller;	3426

activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller::SubscriptionInfoMarshaller::wireformat::marshal::v2::ConsumerControl  
 3627 1376  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller::TransactionIdMarshaller::wireformat::marshal::v2::ConsumerIdMarsh  
 3768 1404  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller::TransactionIdMarshaller::wireformat::marshal::v2::ConsumerInfoMarsh  
 3796 1437  
 activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller::WireFormatMarshaller::wireformat::marshal::v2::ControlCommandMarsh  
 3941 1465  
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::wireformat::marshal::v2::DataArrayResponseMarshaller  
 3979 1498  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBinaryMessageMarshaller::wireformat::marshal::v2::DataResponseMarshaller  
 192 1563  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBinaryMessageMarshaller::wireformat::marshal::v2::DestinationInfoMarshaller  
 243 1698  
 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::wireformat::marshal::v2::DiscoveryEventManager  
 322 1731  
 activemq::wireformat::openwire::marshal::v2::ActiveMQErrorMessageMarshaller::wireformat::marshal::v2::ExceptionResponseMarshaller  
 363 1811  
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::wireformat::marshal::v2::FlushCommandMarshaller  
 389 1910  
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::wireformat::marshal::v2::IntegerResponseMarshaller  
 435 2063  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::wireformat::marshal::v2::JournalQueueAckMarshaller  
 479 2126  
 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::wireformat::marshal::v2::JournalTopicAckMarshaller  
 542 2154  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::wireformat::marshal::v2::JournalTraceMarshaller  
 568 2177  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::wireformat::marshal::v2::JournalTransactionMarshaller  
 597 2208  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::wireformat::marshal::v2::KeepAliveInfoMarshaller  
 625 2235  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::wireformat::marshal::v2::LastPartialCommandMarshaller  
 658 2273  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::wireformat::marshal::v2::LocalTransactionMarshaller  
 686 2317  
 activemq::wireformat::openwire::marshal::v2::BaseQueueInfoMarshaller::wireformat::marshal::v2::MessageAckMarshaller  
 767 2533  
 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::wireformat::marshal::v2::MessageDispatchMarshaller  
 854 2569  
 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::wireformat::marshal::v2::MessageDispatchMarshaller  
 885 2601  
 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::wireformat::marshal::v2::MessageIdMarshaller  
 1264 2630  
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::wireformat::marshal::v2::MessageMarshaller  
 1272 2664  
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::wireformat::marshal::v2::MessagePullMarshaller  
 1303 2702  
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::wireformat::marshal::v2::NetworkBridgeFileMarshaller  
 1333 2752

activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	2877	ActiveMQWireFormatMarshaller,	581	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller,	2990
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	2990	ActiveMQWireFormatMarshaller,	609	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller,	3021
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3021	ActiveMQWireFormatMarshaller,	638	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller,	3054
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3054	ActiveMQWireFormatMarshaller,	666	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller,	3144
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3144	ActiveMQWireFormatMarshaller,	733	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller,	3180
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3180	ActiveMQWireFormatMarshaller,	834	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,	3207
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3207	ActiveMQWireFormatMarshaller,	865	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller,	3244
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3244	ActiveMQWireFormatMarshaller,	1244	activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller,	3327
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3327	ActiveMQWireFormatMarshaller,	1276	activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller,	3370
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3370	ActiveMQWireFormatMarshaller,	1307	activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller,	3422
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3422	ActiveMQWireFormatMarshaller,	1337	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller,	3643
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3643	ActiveMQWireFormatMarshaller,	1380	activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller,	3772
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3772	ActiveMQWireFormatMarshaller,	1408	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,	3812
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3812	ActiveMQWireFormatMarshaller,	1441	activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller,	3933
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3933	ActiveMQWireFormatMarshaller,	1469	activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller,	3971
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,	3971	ActiveMQWireFormatMarshaller,	1502	activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller,	180
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	180	ActiveMQBlobMessageMarshaller,	1568	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller,	223
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,	223	ActiveMQBlobMessageMarshaller,	1702	activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller,	306
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	306	ActiveMQDestinationMarshaller,	1735	activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller,	347
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	347	ActiveMQDestinationMarshaller,	1815	activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller,	373
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	373	ActiveMQDestinationMarshaller,	1914	activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller,	419
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	419	ActiveMQDestinationMarshaller,	2067	activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller,	463
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	463	ActiveMQDestinationMarshaller,	2134	activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller,	525
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	525	ActiveMQDestinationMarshaller,	2158	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller,	553
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	553	ActiveMQDestinationMarshaller,	2181	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,	

activemq::wireformat::openwire::marshal::v3::ActiveMQTransactionalMessage 3945  
 2212  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 3983  
 2239  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 3983  
 2269  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 188  
 2321  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 231  
 2537  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 314  
 2573  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 355  
 2606  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 381  
 2642  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 427  
 2659  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 471  
 2710  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 534  
 2764  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 560  
 2885  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 589  
 2998  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 613  
 3029  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 642  
 3066  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 670  
 3152  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 740  
 3176  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 838  
 3211  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 869  
 3253  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 1248  
 3343  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 1280  
 3366  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 1311  
 3434  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 1341  
 3623  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 1384  
 3776  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 1412  
 3800  
 activemq::wireformat::openwire::marshal::v3::ActiveMQWireFormat 1445

activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatMarshaller,	3050
1473	
activemq::wireformat::openwire::marshal::v4::ActiveMQResponseMarshaller,	3164
1507	
activemq::wireformat::openwire::marshal::v4::ActiveResponseMarshaller,	3192
1572	
activemq::wireformat::openwire::marshal::v4::ActiveWireFormatMarshaller,	3199
1706	
activemq::wireformat::openwire::marshal::v4::DiscoveryLevelMarshaller,	3239
1739	
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller,	3331
1823	
activemq::wireformat::openwire::marshal::v4::FlushQueueFormatMarshaller,	3374
1918	
activemq::wireformat::openwire::marshal::v4::HeaderResponseMarshaller,	3438
2071	
activemq::wireformat::openwire::marshal::v4::InactiveQueueAckMarshaller,	3635
2138	
activemq::wireformat::openwire::marshal::v4::InactiveTopicAckMarshaller,	3780
2166	
activemq::wireformat::openwire::marshal::v4::InactiveTraceMarshaller,	3808
2189	
activemq::wireformat::openwire::marshal::v4::InactiveWireFormatMarshaller,	3937
2220	
activemq::wireformat::openwire::marshal::v4::KeepAliveMarshaller,	3975
2243	
activemq::wireformat::openwire::marshal::v4::LocalPagingFormatMarshaller,	196
2281	
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller,	235
2329	
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller,	318
2541	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller,	359
2581	
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller,	385
2610	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller,	431
2634	
activemq::wireformat::openwire::marshal::v4::MessageMarshaller,	475
2668	
activemq::wireformat::openwire::marshal::v4::MessageRuleMarshaller,	538
2714	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,	564
2768	
activemq::wireformat::openwire::marshal::v4::PersistentWireFormatMarshaller,	593
2890	
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller,	621
2994	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller,	650
3025	
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller,	
3050	
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller,	
3164	
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller,	
3192	
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller,	
3199	
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller,	
3239	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,	
3331	
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller,	
3374	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller,	
3438	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,	
3635	
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,	
3780	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller,	
3808	
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,	
3937	
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller,	
3975	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller,	
196	
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller,	
235	
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,	
318	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller,	
359	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller,	
385	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,	
431	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller,	
475	
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller,	
538	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller,	
564	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller,	
593	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller,	
621	
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller,	
650	

activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, wire::marshal::v5::LocalTransaction  
 678 2325  
 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, wire::marshal::v5::MessageAckMarsh  
 753 2549  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::MessageDispatch  
 846 2577  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::MessageDispatch  
 877 2618  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::MessageIdMarsh  
 1256 2638  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::MessageMarsh  
 1288 2655  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::MessagePullMarsh  
 1319 2706  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::NetworkBridgeFil  
 1349 2760  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::PartialCommand  
 1392 2881  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::ProducerAckMarsh  
 1420 3002  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::ProducerIdMarsh  
 1453 3033  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::ProducerInfoMarsh  
 1481 3062  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::RemoveInfoMarsh  
 1515 3160  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::RemoveSubscrip  
 1555 3188  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::ReplayCommand  
 1718 3219  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::ResponseMarsh  
 1747 3248  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::SessionIdMarsh  
 1819 3339  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::SessionInfoMarsh  
 1926 3358  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::ShutdownInfoMar  
 2079 3430  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::SubscriptionInfoM  
 2130 3631  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::TransactionIdMar  
 2150 3765  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::TransactionInfoM  
 2197 3791  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::WireFormatInfoM  
 2216 3925  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v5::XATransactionIdM  
 2247 3987  
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, wire::marshal::v6::ActiveMQBlobMe  
 2277 200

activemq::wireformat::openwire::marshal::v6::ActiveMQByteMessageMarshaller,	1714
239	
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller,	1727
326	
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller,	1807
367	
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller,	1906
393	
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller,	2059
439	
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller,	2122
483	
activemq::wireformat::openwire::marshal::v6::ActiveMQSerializedMessageMarshaller,	2162
546	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller,	2185
572	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller,	2204
601	
activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller,	2231
629	
activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller,	2265
654	
activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller,	2313
682	
activemq::wireformat::openwire::marshal::v6::BaseCommandMarshaller,	2528
760	
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::MessageDispatchMarshaller,	2589
850	
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::MessageDispatchNotificationMarshaller,	2597
881	
activemq::wireformat::openwire::marshal::v6::ActiveMQControlMarshaller,	2646
1260	
activemq::wireformat::openwire::marshal::v6::ActiveMQErrorMarshaller,	2677
1292	
activemq::wireformat::openwire::marshal::v6::ActiveMQInfoMarshaller,	2722
1323	
activemq::wireformat::openwire::marshal::v6::ActiveMQInfoMarshaller,	2756
1353	
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueControlMarshaller,	2872
1396	
activemq::wireformat::openwire::marshal::v6::ActiveMQWireMarshaller,	3006
1424	
activemq::wireformat::openwire::marshal::v6::ActiveMQWireMarshaller,	3037
1457	
activemq::wireformat::openwire::marshal::v6::ActiveMQWireFormat::openwire::marshal::v6::ProducerInfoMarshaller,	3070
1485	
activemq::wireformat::openwire::marshal::v6::DataArrayResponseMarshaller,	3148
1519	
activemq::wireformat::openwire::marshal::v6::DataResponseMarshaller,	3184
1559	

activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller, 3215  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller, 3262  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller, 3335  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller, 3354  
 activemq::wireformat::openwire::marshal::v6::ShutdownInfoMarshaller, 3418  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller, 3639  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3783  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3804  
 activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3929  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 3967  
 tightMarshal2, 1450  
 activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 782  
 activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1613  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBinaryMessageMarshaller, 185  
 activemq::wireformat::openwire::marshal::v1::ActiveMQByteMessageMarshaller, 227  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 311  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 351  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 378  
 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 424  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 467  
 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 530  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 557  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 585  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 618  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 647  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 2286



2333	243
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller,
2545	323
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller,
2585	363
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller,
2614	390
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller,
2651	436
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller,
2673	479
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller,
2719	542
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller,
2772	569
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller,
2895	597
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller,
3011	626
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller,
3042	659
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller,
3059	687
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller,	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller,
3156	768
activemq::wireformat::openwire::marshal::v1::ActiveMQSubscriptionMarshaller,	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller,
3172	855
activemq::wireformat::openwire::marshal::v1::ActiveMQBrokerInfoMarshaller,	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller,
3204	886
activemq::wireformat::openwire::marshal::v1::ActiveMQConnectionControlMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller,
3258	1265
activemq::wireformat::openwire::marshal::v1::ActiveMQConnectionErrorMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller,
3347	1273
activemq::wireformat::openwire::marshal::v1::ActiveMQConnectionIdMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller,
3363	1303
activemq::wireformat::openwire::marshal::v1::ActiveMQConnectionInfoMarshaller,	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller,
3427	1333
activemq::wireformat::openwire::marshal::v1::ActiveMQConsumerControlMarshaller,	activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller,
3627	1376
activemq::wireformat::openwire::marshal::v1::ActiveMQConsumerIdMarshaller,	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller,
3769	1405
activemq::wireformat::openwire::marshal::v1::ActiveMQConsumerInfoMarshaller,	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller,
3796	1437
activemq::wireformat::openwire::marshal::v1::ActiveMQControlCommandMarshaller,	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller,
3941	1465
activemq::wireformat::openwire::marshal::v1::ActiveMQDataArrayResponseMarshaller,	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller,
3979	1499
activemq::wireformat::openwire::marshal::v2::ActiveMQDataResponseMarshaller,	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller,
193	1564
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationInfoMarshaller,	activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller,

1699	3208
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3244
1732	3244
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3327
1812	3327
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3371
1910	3371
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3423
2064	3423
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3643
2126	3643
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3773
2154	3773
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3812
2177	3812
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3933
2208	3933
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	3971
2236	3971
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	180
2274	180
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	223
2317	223
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	307
2533	307
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	347
2569	347
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	374
2602	374
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	419
2631	419
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	463
2664	463
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	526
2703	526
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	553
2752	553
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	581
2877	581
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	610
2991	610
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	638
3022	638
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	667
3055	667
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	734
3144	734
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	835
3181	835
activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatOpenWireMarshaler	

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

382	1918
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller
428	2072
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller
471	2138
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller
534	2166
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller
561	2189
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller
589	2220
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller
614	2244
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller
643	2282
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v4::LocalTransactionMarshaller
671	2329
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller
741	2541
activemq::wireformat::openwire::marshal::v4::BaseWireFormat	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller
839	2581
activemq::wireformat::openwire::marshal::v4::BaseWireFormat	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller
870	2610
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller
1249	2635
activemq::wireformat::openwire::marshal::v4::ConnectionFormatMarshaller	activemq::wireformat::openwire::marshal::v4::MessageMarshaller
1281	2669
activemq::wireformat::openwire::marshal::v4::ConnectionFormatMarshaller	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller
1311	2715
activemq::wireformat::openwire::marshal::v4::ConnectionFormatMarshaller	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFileMarshaller
1342	2768
activemq::wireformat::openwire::marshal::v4::ConnectionFormatMarshaller	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller
1385	2890
activemq::wireformat::openwire::marshal::v4::ConnectionFormatMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller
1413	2995
activemq::wireformat::openwire::marshal::v4::ConnectionFormatMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller
1446	3026
activemq::wireformat::openwire::marshal::v4::ConnectionFormatMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller
1474	3050
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller
1507	3164
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionMarshaller
1572	3193
activemq::wireformat::openwire::marshal::v4::DestinationFormatMarshaller	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
1707	3200
activemq::wireformat::openwire::marshal::v4::DiscoveryFormatMarshaller	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller
1740	3240
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller
1824	3331
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller

3375	1319
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller,
3439	1350
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller,
3635	1393
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller,
3780	1421
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller,
3808	1454
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller,
3937	1482
activemq::wireformat::openwire::marshal::v4::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller,
3975	1515
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller,
197	1556
activemq::wireformat::openwire::marshal::v5::ActiveMQBinaryMessageMarshaller	activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller,
235	1719
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller,
319	1748
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller,
359	1820
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller,
386	1926
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller,
432	2080
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueFormatMarshaller	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller,
475	2130
activemq::wireformat::openwire::marshal::v5::ActiveMQSimpleMessageMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller,
538	2150
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller,
565	2197
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller,
593	2216
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller,
622	2248
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller,
651	2278
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller,
679	2325
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller,
755	2549
activemq::wireformat::openwire::marshal::v5::BrokenWireFormat	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller,
847	2577
activemq::wireformat::openwire::marshal::v5::BrokenWireFormat	activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller,
878	2619
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller,
1257	2639
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v5::MessageMarshaller,
1289	2656
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller,

2707	546
activemq::wireformat::openwire::marshal::v5::ActiveMQBridgeFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ActiveMQTempD
2760	573
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ActiveMQTempQ
2882	601
activemq::wireformat::openwire::marshal::v5::ActiveMQAddFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ActiveMQTempTo
3003	630
activemq::wireformat::openwire::marshal::v5::ActiveMQDeleteFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ActiveMQTextMe
3034	655
activemq::wireformat::openwire::marshal::v5::ActiveMQInfoFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ActiveMQTopicM
3063	683
activemq::wireformat::openwire::marshal::v5::ActiveMQInfoFormatMarshaller	activemq::wireformat::openwire::marshal::v6::BaseCommandM
3160	761
activemq::wireformat::openwire::marshal::v5::ActiveMQSubscriptionInfoFormatMarshaller	activemq::wireformat::openwire::marshal::v6::BrokerIdMarshall
3189	851
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::BrokerInfoMarsha
3220	882
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionContr
3249	1261
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionErrorM
3339	1293
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionIdMar
3359	1323
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ConnectionInfoM
3431	1354
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ConsumerContro
3631	1397
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ConsumerIdMars
3765	1425
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ConsumerInfoMa
3792	1458
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::ControlCommand
3925	1486
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::DataArrayRespon
3987	1519
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatMarshaller	activemq::wireformat::openwire::marshal::v6::DataResponseM
201	1560
activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageMarshaller	activemq::wireformat::openwire::marshal::v6::DestinationInfoM
239	1715
activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller	activemq::wireformat::openwire::marshal::v6::DiscoveryEventM
327	1728
activemq::wireformat::openwire::marshal::v6::ActiveMQMapMessageMarshaller	activemq::wireformat::openwire::marshal::v6::ExceptionRespon
367	1807
activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller	activemq::wireformat::openwire::marshal::v6::FlushCommandM
394	1906
activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller	activemq::wireformat::openwire::marshal::v6::IntegerResponse
440	2060
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueFormatMarshaller	activemq::wireformat::openwire::marshal::v6::JournalQueueAck
483	2122
activemq::wireformat::openwire::marshal::v6::ActiveMQQueueFormatMarshaller	activemq::wireformat::openwire::marshal::v6::JournalTopicAckM

2162  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller,  
 2185  
 activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller,  
 2204  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller,  
 2231  
 activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller,  
 2265  
 activemq::wireformat::openwire::marshal::v6::LocalTransactionIdMarshaller,  
 2313  
 activemq::wireformat::openwire::marshal::v6::MessageAckMarshaller,  
 2529  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller,  
 2589  
 activemq::wireformat::openwire::marshal::v6::MessageDispatchNotificationMarshaller,  
 2598  
 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,  
 2647  
 activemq::wireformat::openwire::marshal::v6::MessageIdMarshaller,  
 2677  
 activemq::wireformat::openwire::marshal::v6::MessagePullMarshaller,  
 2723  
 activemq::wireformat::openwire::marshal::v6::MessageBridgeFilterMarshaller,  
 2756  
 activemq::wireformat::openwire::marshal::v6::MessageNestedObjectMarshaller,  
 2873  
 activemq::wireformat::openwire::marshal::v6::ProducerAckMarshaller,  
 3007  
 activemq::wireformat::openwire::marshal::v6::ProducerIdMarshaller,  
 3038  
 activemq::wireformat::openwire::marshal::v6::ProducerInfoMarshaller,  
 3071  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller,  
 3148  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,  
 3185  
 activemq::wireformat::openwire::marshal::v6::ReplyCommandMarshaller,  
 3216  
 activemq::wireformat::openwire::marshal::v6::ResponseMarshaller,  
 3263  
 activemq::wireformat::openwire::marshal::v6::SessionIdMarshaller,  
 3335  
 activemq::wireformat::openwire::marshal::v6::SessionInfoMarshaller,  
 3355  
 activemq::wireformat::openwire::marshal::v6::ShutdownWireMarshaller,  
 3419  
 activemq::wireformat::openwire::marshal::v6::SubscriptionInfoMarshaller,  
 3639  
 activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller,

activemq::wireformat::openwire::marshal::DataStreamWireFormat::openwire::marshal::v1::DataArrayResponse  
 1620 1511  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBinaryMessageMarshaller::marshal::v1::DataResponseMarshaller  
 185 1576  
 activemq::wireformat::openwire::marshal::v1::ActiveMQBinaryMessageMarshaller::marshal::v1::DestinationInfoMarshaller  
 228 1711  
 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::marshal::v1::DiscoveryEventManager  
 311 1744  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::marshal::v1::ExceptionResponseMarshaller  
 352 1828  
 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::marshal::v1::FlushCommandMarshaller  
 378 1922  
 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::marshal::v1::IntegerResponseMarshaller  
 424 2076  
 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::openwire::marshal::v1::JournalQueueAckMarshaller  
 467 2142  
 activemq::wireformat::openwire::marshal::v1::ActiveMQSerializedMessageMarshaller::marshal::v1::JournalTopicAckMarshaller  
 530 2171  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::marshal::v1::JournalTraceMarshaller  
 558 2193  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::marshal::v1::JournalTransactionMarshaller  
 586 2225  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::marshal::v1::KeepAliveInfoMarshaller  
 618 2252  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::marshal::v1::LastPartialCommandMarshaller  
 647 2286  
 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::openwire::marshal::v1::LocalTransactionMarshaller  
 675 2334  
 activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::openwire::marshal::v1::MessageAckMarshaller  
 749 2545  
 activemq::wireformat::openwire::marshal::v1::BinaryWireFormat::openwire::marshal::v1::MessageDispatchMarshaller  
 843 2586  
 activemq::wireformat::openwire::marshal::v1::BinaryWireFormat::openwire::marshal::v1::MessageDispatchMarshaller  
 874 2615  
 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::marshal::v1::MessageIdMarshaller  
 1253 2651  
 activemq::wireformat::openwire::marshal::v1::ConnectionFormatMarshaller::openwire::marshal::v1::MessageMarshaller  
 1285 2674  
 activemq::wireformat::openwire::marshal::v1::ConnectionMarshaller::openwire::marshal::v1::MessagePullMarshaller  
 1316 2719  
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::openwire::marshal::v1::NetworkBridgeFileMarshaller  
 1346 2772  
 activemq::wireformat::openwire::marshal::v1::ConnectionMarshaller::openwire::marshal::v1::PartialCommandMarshaller  
 1389 2895  
 activemq::wireformat::openwire::marshal::v1::ConnectionMarshaller::openwire::marshal::v1::ProducerAckMarshaller  
 1417 3011  
 activemq::wireformat::openwire::marshal::v1::ConnectionMarshaller::openwire::marshal::v1::ProducerIdMarshaller  
 1450 3042  
 activemq::wireformat::openwire::marshal::v1::ConnectionMarshaller::openwire::marshal::v1::ProducerInfoMarshaller  
 1478 3059



activemq::wireformat::openwire::marshal::v1::ActiveMQInfoMarshaller,	3156	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller,	769
activemq::wireformat::openwire::marshal::v1::ActiveMQSubscriptionInfoMarshaller,	3173	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller,	855
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3204	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller,	886
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3259	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller,	1265
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3347	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller,	1273
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3363	activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller,	1304
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3427	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller,	1334
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3627	activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller,	1377
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3769	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller,	1405
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3796	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller,	1438
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3942	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller,	1466
activemq::wireformat::openwire::marshal::v1::ActiveMQWireFormatMarshaller,	3980	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller,	1499
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller,	193	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller,	1564
activemq::wireformat::openwire::marshal::v2::ActiveMQByteMessageMarshaller,	244	activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller,	1699
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller,	323	activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller,	1732
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller,	364	activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller,	1812
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller,	390	activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller,	1910
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller,	436	activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller,	2064
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller,	479	activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller,	2126
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller,	542	activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller,	2155
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller,	569	activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller,	2177
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller,	598	activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller,	2209
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller,	626	activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller,	2236
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller,	659	activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller,	2274
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller,	687	activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller,	2318

activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller,openwire::marshal::v3::ActiveMQDestinationMarshaller 2533  
 activemq::wireformat::openwire::marshal::v2::ActiveMQDispatcher,openwire::marshal::v3::ActiveMQDispatcher 2570  
 activemq::wireformat::openwire::marshal::v2::ActiveMQDispatchOptionsMarshaller,openwire::marshal::v3::ActiveMQDispatchOptionsMarshaller 2602  
 activemq::wireformat::openwire::marshal::v2::ActiveMQIDMarshaller,openwire::marshal::v3::ActiveMQIDMarshaller 2631  
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller,openwire::marshal::v3::ActiveMQQueueMarshaller 2665  
 activemq::wireformat::openwire::marshal::v2::ActiveMQRuleMarshaller,openwire::marshal::v3::ActiveMQRuleMarshaller 2703  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBridgeFormatMarshaller,openwire::marshal::v3::ActiveMQBridgeFormatMarshaller 2752  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireFormatMarshaller,openwire::marshal::v3::ActiveMQWireFormatMarshaller 2878  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller,openwire::marshal::v3::ActiveMQTempDestinationMarshaller 2991  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMarshaller,openwire::marshal::v3::ActiveMQTextMarshaller 3022  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller,openwire::marshal::v3::ActiveMQTopicMarshaller 3055  
 activemq::wireformat::openwire::marshal::v2::ActiveMQWireMarshaller,openwire::marshal::v3::ActiveMQWireMarshaller 3144  
 activemq::wireformat::openwire::marshal::v2::ActiveMQSubscriptionInfoMarshaller,openwire::marshal::v3::ActiveMQSubscriptionInfoMarshaller 3181  
 activemq::wireformat::openwire::marshal::v2::ActiveMQBrokerInfoMarshaller,openwire::marshal::v3::ActiveMQBrokerInfoMarshaller 3208  
 activemq::wireformat::openwire::marshal::v2::ActiveMQResponseMarshaller,openwire::marshal::v3::ActiveMQResponseMarshaller 3245  
 activemq::wireformat::openwire::marshal::v2::ActiveMQErrorMarshaller,openwire::marshal::v3::ActiveMQErrorMarshaller 3327  
 activemq::wireformat::openwire::marshal::v2::ActiveMQConnectionIdMarshaller,openwire::marshal::v3::ActiveMQConnectionIdMarshaller 3371  
 activemq::wireformat::openwire::marshal::v2::ActiveMQConnectionInfoMarshaller,openwire::marshal::v3::ActiveMQConnectionInfoMarshaller 3423  
 activemq::wireformat::openwire::marshal::v2::ActiveMQSubscriptionInfoMarshaller,openwire::marshal::v3::ActiveMQSubscriptionInfoMarshaller 3643  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionMarshaller,openwire::marshal::v3::ActiveMQTransactionMarshaller 3773  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTransactionInfoMarshaller,openwire::marshal::v3::ActiveMQTransactionInfoMarshaller 3812  
 activemq::wireformat::openwire::marshal::v2::ActiveMQFormatMarshaller,openwire::marshal::v3::ActiveMQFormatMarshaller 3934  
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller,openwire::marshal::v3::ActiveMQTempDestinationMarshaller 3972  
 activemq::wireformat::openwire::marshal::v3::ActiveMQBinaryMessageMarshaller,openwire::marshal::v3::ActiveMQBinaryMessageMarshaller 181  
 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationInfoMarshaller,openwire::marshal::v3::ActiveMQDestinationInfoMarshaller 224

activemq::wireformat::openwire::marshal::v3::DiscoveryErrorMarshaller,	1736
activemq::wireformat::openwire::marshal::v3::DiscoveryResponseMarshaller,	1816
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller,	1914
activemq::wireformat::openwire::marshal::v3::FlushQueueFormatMarshaller,	2068
activemq::wireformat::openwire::marshal::v3::HeaderResponseMarshaller,	2134
activemq::wireformat::openwire::marshal::v3::HeaderQueueFormatMarshaller,	2159
activemq::wireformat::openwire::marshal::v3::IllegalResponseMarshaller,	2181
activemq::wireformat::openwire::marshal::v3::InvalidQueueIdMarshaller,	2213
activemq::wireformat::openwire::marshal::v3::InvalidTopicIdMarshaller,	2240
activemq::wireformat::openwire::marshal::v3::InvalidTransactionIdMarshaller,	2270
activemq::wireformat::openwire::marshal::v3::InvalidXidMarshaller,	2322
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller,	2537
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller,	2574
activemq::wireformat::openwire::marshal::v3::MessageDispatchInfoMarshaller,	2607
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	2643
activemq::wireformat::openwire::marshal::v3::MessageMarshaller,	2661
activemq::wireformat::openwire::marshal::v3::MessageRuleMarshaller,	2711
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller,	2764
activemq::wireformat::openwire::marshal::v3::PersistentQueueIdMarshaller,	2886
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller,	2999
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller,	3030
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller,	3067
activemq::wireformat::openwire::marshal::v3::RemoveQueueMarshaller,	3152
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller,	3177
activemq::wireformat::openwire::marshal::v3::ReplyQueueFormatMarshaller,	3212
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller,	3254
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller,	3343
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller,	3367
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller,	3435
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller,	3623
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller,	3777
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller,	3800
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller,	3946
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller,	3984
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller,	189
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller,	232
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller,	315
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller,	356
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller,	382
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller,	428
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller,	471
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller,	534
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller,	562
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller,	590
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller,	614
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller,	643
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller,	671
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller,	742
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller,	839
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller,	870

activemq::wireformat::openwire::marshal::v4::ActiveQueueControlMarshaller; marshal::v4::MessageIdMarshaller  
 1249 2635  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::MessageMarshaller  
 1281 2669  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::MessagePullMarshaller  
 1312 2715  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::NetworkBridgeFileMarshaller  
 1342 2768  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::PartialCommandMarshaller  
 1385 2891  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::ProducerAckMarshaller  
 1413 2995  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::ProducerIdMarshaller  
 1446 3026  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::ProducerInfoMarshaller  
 1474 3051  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::RemoveInfoMarshaller  
 1507 3164  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::RemoveSubscriptionMarshaller  
 1572 3193  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::ReplayCommandMarshaller  
 1707 3200  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::ResponseMarshaller  
 1740 3240  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::SessionIdMarshaller  
 1824 3331  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::SessionInfoMarshaller  
 1918 3375  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::ShutdownInfoMarshaller  
 2072 3439  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::SubscriptionInfoMarshaller  
 2138 3635  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::TransactionIdMarshaller  
 2167 3781  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::TransactionInfoMarshaller  
 2189 3808  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::WireFormatInfoMarshaller  
 2221 3938  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v4::XATransactionIdMarshaller  
 2244 3976  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v5::ActiveMQBlobMarshaller  
 2282 197  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v5::ActiveMQBytesMarshaller  
 2330 236  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v5::ActiveMQDestinationMarshaller  
 2541 319  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v5::ActiveMQMapMarshaller  
 2582 360  
 activemq::wireformat::openwire::marshal::v4::ActiveQueueFormatMarshaller; marshal::v5::ActiveMQMessageMarshaller  
 2611 386

activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller,	2080
432	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueFormatOpenWire::marshal::v5::JournalQueueAckMarshaller,	2130
475	
activemq::wireformat::openwire::marshal::v5::ActiveMQSimpleMessageMarshaller,	2151
538	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller,	2197
565	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller,	2217
594	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller;	2248
622	marshal::v5::KeepAliveInfoMarshaller,
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller,	2278
651	
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller,	2326
679	
activemq::wireformat::openwire::marshal::v5::ActiveMQwireIdMarshaller,	2549
756	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::MessageDispatchMarshaller,	2578
847	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::MessageDispatchNotificationM	2619
878	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::MessageIdMarshaller,	2639
1257	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::MessageMarshaller,	2656
1289	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::MessagePullMarshaller,	2707
1320	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::NetworkBridgeFilterMarshaller,	2760
1350	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::PartialCommandMarshaller,	2882
1393	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::ProducerAckMarshaller,	3003
1421	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::ProducerIdMarshaller,	3034
1454	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::ProducerInfoMarshaller,	3063
1482	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::RemoveInfoMarshaller,	3160
1515	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller,	3189
1556	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::ReplayCommandMarshaller,	3220
1719	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::ResponseMarshaller,	3249
1748	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::SessionIdMarshaller,	3339
1820	
activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormat::openwire::marshal::v5::SessionInfoMarshaller,	3359
1926	

activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ConnectionInfoMarshaller 1354  
 3431  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ConsumerControlMarshaller 1397  
 3631  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ConsumerIdMarshaller 1425  
 3766  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ConsumerInfoMarshaller 1458  
 3792  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::ControlCommandMarshaller 1486  
 3926  
 activemq::wireformat::openwire::marshal::v5::ActiveMQWireFormatOpenwire::marshal::v6::DataArrayResponseMarshaller 1519  
 3988  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageMarshaller 1560  
 201  
 activemq::wireformat::openwire::marshal::v6::ActiveMQBinaryMessageUnmarshaller 1715  
 240  
 activemq::wireformat::openwire::marshal::v6::ActiveMQDestinationMarshaller 1728  
 327  
 activemq::wireformat::openwire::marshal::v6::ActiveMQErrorMessageMarshaller 1808  
 368  
 activemq::wireformat::openwire::marshal::v6::ActiveMQMessageMarshaller 1906  
 394  
 activemq::wireformat::openwire::marshal::v6::ActiveMQObjectMessageMarshaller 2060  
 440  
 activemq::wireformat::openwire::marshal::v6::ActiveMQQueueMarshaller 2122  
 483  
 activemq::wireformat::openwire::marshal::v6::ActiveMQSimpleMessageMarshaller 2163  
 546  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempDestinationMarshaller 2185  
 573  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempQueueMarshaller 2205  
 602  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTempTopicMarshaller 2232  
 630  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTextMessageMarshaller 2266  
 655  
 activemq::wireformat::openwire::marshal::v6::ActiveMQTopicMarshaller 2314  
 683  
 activemq::wireformat::openwire::marshal::v6::BaseQueueWireFormatOpenwire::marshal::v6::MessageAckMarshaller 2529  
 763  
 activemq::wireformat::openwire::marshal::v6::BrokerWireFormatOpenwire::marshal::v6::MessageDispatcherMarshaller 2590  
 851  
 activemq::wireformat::openwire::marshal::v6::BrokerWireFormatOpenwire::marshal::v6::MessageDispatcherUnmarshaller 2598  
 882  
 activemq::wireformat::openwire::marshal::v6::ConnectionControlMarshaller 2647  
 1261  
 activemq::wireformat::openwire::marshal::v6::ConnectionErrorMarshaller 2678  
 1293  
 activemq::wireformat::openwire::marshal::v6::ConnectionIdMarshaller 2723  
 1324

Generated on Sun Jan 8 2012 23:14:17 for activemq-cpp-3.2.5 by Doxygen

- decaf::lang::Long, 2389
- toByteArray
  - decaf::io::ByteArrayOutputStream, 994
- toDays
  - decaf::util::concurrent::TimeUnit, 3753
- toDegrees
  - decaf::lang::Math, 2471
- toDestinationOption
  - activemq::core::ActiveMQConstants, 282
- toHexFromBytes
  - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 791
- toHexString
  - decaf::lang::Double, 1760
  - decaf::lang::Float, 1874
  - decaf::lang::Integer, 2050
  - decaf::lang::Long, 2389
- toHours
  - decaf::util::concurrent::TimeUnit, 3754
- toMicros
  - decaf::util::concurrent::TimeUnit, 3754
- toMillis
  - decaf::util::concurrent::TimeUnit, 3754
- toMinutes
  - decaf::util::concurrent::TimeUnit, 3755
- toNanos
  - decaf::util::concurrent::TimeUnit, 3755
- toOctalString
  - decaf::lang::Integer, 2051
  - decaf::lang::Long, 2390
- TOPIC
  - cms::Destination, 1689
- TOPIC\_PREFIX
  - activemq::wireformat::stomp::StompCommandConstants, 3576
- TOPIC\_QUALIFIED\_PREFIX
  - activemq::commands::ActiveMQDestination, 304
- toRadians
  - decaf::lang::Math, 2471
- toSeconds
  - decaf::util::concurrent::TimeUnit, 3756
- toStream
  - activemq::wireformat::stomp::StompFrame, 3581
- toString
  - activemq::commands::ActiveMQBlobMessage, 177
  - activemq::commands::ActiveMQBytesMessage, 214
  - activemq::commands::ActiveMQDestination, 302
  - activemq::commands::ActiveMQMapMessage, 344
  - activemq::commands::ActiveMQMessage, 370
  - activemq::commands::ActiveMQObjectMessage, 416
  - activemq::commands::ActiveMQQueue, 456
  - activemq::commands::ActiveMQStreamMessage, 517
  - activemq::commands::ActiveMQTempDestination, 550
  - activemq::commands::ActiveMQTempQueue, 578
  - activemq::commands::ActiveMQTempTopic, 606
  - activemq::commands::ActiveMQTextMessage, 635
  - activemq::commands::ActiveMQTopic, 663
  - activemq::commands::BaseCommand, 729
  - activemq::commands::BaseDataStructure, 796
  - activemq::commands::BooleanExpression, 817
  - activemq::commands::BrokerId, 831
  - activemq::commands::BrokerInfo, 861
  - activemq::commands::Command, 1169
  - activemq::commands::ConnectionControl, 1240
  - activemq::commands::ConnectionError, 1269
  - activemq::commands::ConnectionId, 1300
  - activemq::commands::ConnectionInfo, 1329
  - activemq::commands::ConsumerControl, 1372
  - activemq::commands::ConsumerId, 1401
  - activemq::commands::ConsumerInfo, 1432
  - activemq::commands::ControlCommand, 1461
  - activemq::commands::DataArrayResponse, 1495
  - activemq::commands::DataResponse, 1552
  - activemq::commands::DataStructure, 214



- 1632
- activemq::commands::DestinationInfo, 1695
- activemq::commands::DiscoveryEvent, 1724
- activemq::commands::ExceptionResponse, 1804
- activemq::commands::FlushCommand, 1902
- activemq::commands::IntegerResponse, 2056
- activemq::commands::JournalQueueAck, 2118
- activemq::commands::JournalTopicAck, 2147
- activemq::commands::JournalTrace, 2174
- activemq::commands::JournalTransaction, 2201
- activemq::commands::KeepAliveInfo, 2227
- activemq::commands::LastPartialCommand, 2262
- activemq::commands::LocalTransactionId, 2310
- activemq::commands::Message, 2490
- activemq::commands::MessageAck, 2525
- activemq::commands::MessageDispatch, 2558
- activemq::commands::MessageDispatchNotification, 2594
- activemq::commands::MessageId, 2627
- activemq::commands::MessagePull, 2698
- activemq::commands::NetworkBridgeFilter, 2748
- activemq::commands::PartialCommand, 2869
- activemq::commands::ProducerAck, 2987
- activemq::commands::ProducerId, 3018
- activemq::commands::ProducerInfo, 3046
- activemq::commands::RemoveInfo, 3140
- activemq::commands::RemoveSubscriptionInfo, 3168
- activemq::commands::ReplayCommand, 3196
- activemq::commands::Response, 3230
- activemq::commands::SessionId, 3324
- activemq::commands::SessionInfo, 3351
- activemq::commands::ShutdownInfo, 3415
- activemq::commands::SubscriptionInfo, 3619
- activemq::commands::TransactionId, 3762
- activemq::commands::TransactionInfo, 3788
- activemq::commands::WireFormatInfo, 3922
- activemq::commands::XATransactionId, 3964
- activemq::core::ActiveMQConstants, 282
- activemq::state::ConnectionState, 1361
- activemq::state::ConsumerState, 1459
- activemq::state::ProducerState, 3072
- activemq::state::SessionState, 3379
- activemq::state::TransactionState, 3814
- activemq::util::ActiveMQProperties, 452
- activemq::util::PrimitiveList, 2940
- activemq::util::PrimitiveMap, 2950
- activemq::util::PrimitiveValueNode, 2974
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 792
- cms::CMSProperties, 1138
- decaf::io::ByteArrayOutputStream, 994
- decaf::io::FilterOutputStream, 1864
- decaf::io::InputStream, 2010
- decaf::io::OutputStream, 2860
- decaf::lang::Boolean, 815
- decaf::lang::Byte, 926
- decaf::lang::Character, 1076
- decaf::lang::CharSequence, 1109
- decaf::lang::Double, 1760
- decaf::lang::Float, 1874
- decaf::lang::Integer, 2051, 2052
- decaf::lang::Long, 2390
- decaf::lang::Short, 3388
- decaf::lang::String, 3612
- decaf::lang::Thread, 3715
- decaf::net::InetAddress, 1981
- decaf::net::ServerSocket, 3301
- decaf::net::Socket, 3463
- decaf::net::SocketImpl, 3480
- decaf::net::URI, 3864
- decaf::nio::ByteBuffer, 1022
- decaf::nio::CharBuffer, 1106
- decaf::nio::DoubleBuffer, 1784
- decaf::nio::FloatBuffer, 1898
- decaf::nio::IntBuffer, 2037
- decaf::nio::LongBuffer, 2414
- decaf::nio::ShortBuffer, 3411
- decaf::security::cert::Certificate, 1057
- decaf::util::concurrent::atomic::AtomicBoolean, 707

decaf::util::concurrent::atomic::AtomicInt, 713	TRANSACTION_STATE_FORGET	activemq::core::ActiveMQConstants, 281
decaf::util::concurrent::atomic::AtomicReference, 718	TRANSACTION_STATE_PREPARE	activemq::core::ActiveMQConstants, 281
decaf::util::concurrent::locks::ReentrantLock, 3131	TRANSACTION_STATE_RECOVER	activemq::core::ActiveMQConstants, 281
decaf::util::concurrent::Semaphore, 3287	TRANSACTION_STATE_ROLLBACK	activemq::core::ActiveMQConstants, 281
decaf::util::concurrent::TimeUnit, 3756	TransactionId	
decaf::util::Date, 1637	activemq::commands::TransactionId, 3760	
decaf::util::logging::Level, 2293	transactionId	
decaf::util::Properties, 3081	activemq::commands::JournalTopicAck, 2147	
decaf::util::UUID, 3906	activemq::commands::JournalTransaction, 2201	
total	activemq::commands::Message, 2492	
inflate_state, 1984	activemq::commands::MessageAck, 2526	
total_in	activemq::commands::TransactionInfo, 3789	
z_stream_s, 3991	TransactionIdMarshaller	
total_out	activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3767	
z_stream_s, 3991	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3771	
toURI	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3775	
activemq::util::CompositeData, 1193	activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3778	
toURIOption	activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3763	
activemq::core::ActiveMQConstants, 282	activemq::wireformat::openwire::marshal::v6::TransactionIdMarshaller, 3782	
toURL	TransactionInfo	
decaf::net::URI, 3864	activemq::commands::TransactionInfo, 3786	
Trace	TransactionInfoMarshaller	
zutil.h, 4439	activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3794	
Tracec	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3810	
zutil.h, 4439	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3798	
Tracecv	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3806	
zutil.h, 4439	activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3790	
Tracev	activemq::wireformat::openwire::marshal::v6::TransactionInfoMarshaller, 3802	
zutil.h, 4439	TransactionState	
Tracevv	activemq::core::ActiveMQConstants, 281	
zutil.h, 4439		
track		
activemq::state::ConnectionStateTracker, 1367		
trackBack		
activemq::state::ConnectionStateTracker, 1367		
Tracked		
activemq::state::Tracked, 3759		
TRANSACTION_STATE_BEGIN		
activemq::core::ActiveMQConstants, 281		
TRANSACTION_STATE_COMMITONEPHASE		
activemq::core::ActiveMQConstants, 281		
TRANSACTION_STATE_COMMITTWOPHASE		
activemq::core::ActiveMQConstants, 281		
TRANSACTION_STATE_END		
activemq::core::ActiveMQConstants, 281		

activemq::state::TransactionState, 3814      zutil.h, 4439  
 transfer      tryAcquire  
     decaf::internal::util::concurrent::TransferQueue, 3287–  
         3816, 3817      3289  
     decaf::internal::util::concurrent::TransferStack  
         3818      activemq::core::MessageDispatchChannel,  
 TransferQueue      2564  
     decaf::internal::util::concurrent::TransferQueue, 3657  
         3816      decaf::io::InputStream, 2011  
 TransferStack      decaf::io::OutputStream, 2860  
     decaf::internal::util::concurrent::TransferStack  
         3818      decaf::util::AbstractCollection, 158  
 TransportFilter      decaf::util::concurrent::ConcurrentStlMap,  
     activemq::transport::TransportFilter, 3829      1217  
 transportInterrupted      decaf::util::concurrent::locks::Lock, 2339,  
     activemq::core::ActiveMQConnection,      2340  
         264      decaf::util::concurrent::locks::ReentrantLock,  
     activemq::state::ConnectionStateTracker,      3131, 3132  
         1367      decaf::util::concurrent::Mutex, 2738  
     activemq::transport::DefaultTransportListener, 3648  
         1671      decaf::util::concurrent::Synchronizable,  
     activemq::transport::failover::FailoverTransportListener, 3554  
         1850      decaf::util::StlQueue, 3562  
     activemq::transport::TransportFilter, 3835  
     activemq::transport::TransportListener,      decaf::internal::util::concurrent::MutexImpl,  
         3837      2743  
 transportResumed      TYPE  
     activemq::core::ActiveMQConnection,      inflate.h, 4424  
         264      type  
     activemq::transport::DefaultTransportListener, 2201  
         1671      activemq::commands::JournalTransaction,  
     activemq::transport::failover::FailoverTransportListener, 2492  
         1850      activemq::commands::Message, 2492  
     activemq::transport::TransportFilter, 3835      activemq::commands::TransactionInfo,  
     activemq::transport::TransportListener, TYPE DO      3789  
         3837      inflate.h, 4424  
 tree\_desc      uch  
     deflate.h, 4421      zutil.h, 4439  
 tree\_desc\_s, 3840      uchf  
     dyn\_tree, 3840      zutil.h, 4439  
     max\_code, 3840      ulnt  
     stat\_desc, 3840      zconf.h, 4429  
 trees.h      ulntf  
     \_dist\_code, 4426      zconf.h, 4430  
     \_length\_code, 4427      ulg  
     base\_dist, 4427      zutil.h, 4439  
     base\_length, 4427      uLong  
     static\_dtree, 4428      zconf.h, 4430  
     static\_ltree, 4428      uLongf  
 TRY\_FREE      uLongf

- zconf.h, 4430
- uncaughtException
  - decaf::lang::Thread::UncaughtExceptionHandler, 2957
  - 3841
- UnknownHostException
  - decaf::net::UnknownHostException, 3842, 2957
  - 3843
- UnknownServiceException
  - decaf::net::UnknownServiceException, 2344
  - 3845, 3846
- unlock
  - activemq::core::MessageDispatchChannel, 3093
  - 2564
  - decaf::internal::util::concurrent::MutexImpl, 3840
  - 2743
  - decaf::internal::util::concurrent::Synchronization, 3950
  - 3657
  - decaf::io::InputStream, 2011
  - decaf::io::OutputStream, 2861
  - decaf::util::AbstractCollection, 159
  - decaf::util::concurrent::ConcurrentStlMap, 3576
  - 1217
  - decaf::util::concurrent::Lock, 2335
  - decaf::util::concurrent::locks::Lock, 2341
  - decaf::util::concurrent::locks::ReentrantLock, 3138
  - 3133
  - decaf::util::concurrent::Mutex, 2739
  - decaf::util::concurrent::Synchronizable, 3650
  - decaf::util::StlMap, 3554
  - decaf::util::StlQueue, 3562
- unmarshal
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2955
  - 2848
  - activemq::wireformat::openwire::utils::BooleanStlMap, 820
  - activemq::wireformat::stomp::StompWireFormat, 3588
  - activemq::wireformat::WireFormat, 3910
- unmarshalList
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2955
- unmarshalMap
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2956
- unmarshalPrimitive
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2956
- unmarshalPrimitiveList
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2957
- unmarshalPrimitiveMap
  - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2957
- unpark
  - decaf::util::concurrent::locks::LockSupport, 2344
- unread
  - decaf::io::PushbackInputStream, 3092, 3093
  - unregisterFactory
  - activemq::transport::TransportRegistry, 3840
  - activemq::wireformat::WireFormatRegistry, 3950
  - unsetenv
  - decaf::lang::System, 3677
  - UNSUBSCRIBE
  - activemq::wireformat::stomp::StompCommandConstants, 3576
  - unsubscribe
  - activemq::cmsutil::PooledSession, 2918
  - activemq::core::ActiveMQSession, 502
  - cms::Session, 3318
  - UnsupportedEncodingException
    - decaf::io::UnsupportedEncodingException, 3847, 3848
  - UnsupportedOperationException
    - cms::UnsupportedOperationException, 3853
    - activemq::core::UnsupportedOperationException, 3850, 3851
  - update
    - decaf::util::zip::Adler32, 692, 693
    - decaf::util::zip::Checksum, 1115, 1116
    - decaf::util::zip::CRC32, 1491, 1492
    - decaf::net::URI, 3855–3857
    - URIEncoderDecoder
      - decaf::internal::net::URIEncoderDecoder, 3866
    - URIHelper
      - decaf::internal::net::URIHelper, 3869
    - URIParser
      - activemq::core::ActiveMQConstants, 281
    - uriParams
      - activemq::core::ActiveMQConstants::StaticInitializer, 3529

---

uriParamsMap	validateUserinfo
activemq::core::ActiveMQConstants::StaticInitializer, 3529	decaf::internal::net::URIHelper, 3874
URIPool	value
activemq::transport::failover::URIPool, 3875	activemq::commands::BrokerId, 832
URISyntaxException	activemq::commands::ConnectionId, 1300
decaf::net::URISyntaxException, 3881, 3882	activemq::commands::ConsumerId, 1402
URIType	activemq::commands::LocalTransactionId, 2310
decaf::internal::net::URIType, 3885	activemq::commands::ProducerId, 3018
URL	activemq::commands::SessionId, 3324
decaf::net::URL, 3893	valueOf
userID	decaf::lang::Boolean, 815
activemq::commands::Message, 2492	decaf::lang::Byte, 926, 927
userName	decaf::lang::Character, 1076
activemq::commands::ConnectionInfo, 1330	decaf::lang::Double, 1761
ush	decaf::lang::Float, 1875
zutil.h, 4439	decaf::lang::Integer, 2052, 2053
ushf	decaf::lang::Long, 2390, 2391
zutil.h, 4440	decaf::lang::Short, 3388, 3389
UTFDataFormatException	decaf::util::concurrent::TimeUnit, 3756
decaf::io::UTFDataFormatException, 3898, 3899	values
UUID	decaf::util::concurrent::ConcurrentStlMap, 1217
decaf::util::UUID, 3902	decaf::util::concurrent::TimeUnit, 3757
	decaf::util::Map, 2430
	decaf::util::StlMap, 3554
val	variant
code, 1154	decaf::util::UUID, 3906
valid	verify
decaf::io::FileDescriptor, 1852	decaf::security::cert::Certificate, 1057, 1058
validate	version
decaf::internal::net::URIEncoderDecoder, 3867	decaf::util::UUID, 3906
validateAuthority	visit
decaf::internal::net::URIHelper, 3872	activemq::commands::BrokerError, 827
validateFragment	activemq::commands::BrokerInfo, 861
decaf::internal::net::URIHelper, 3872	activemq::commands::Command, 1170
validatePath	activemq::commands::ConnectionControl, 1241
decaf::internal::net::URIHelper, 3873	activemq::commands::ConnectionError, 1269
validateQuery	activemq::commands::ConnectionInfo, 1329
decaf::internal::net::URIHelper, 3873	activemq::commands::ConsumerControl, 1372
validateScheme	activemq::commands::ConsumerInfo, 1433
decaf::internal::net::URIHelper, 3873	activemq::commands::ControlCommand, 1462
validateSimple	
decaf::internal::net::URIEncoderDecoder, 3867	
validateSsp	
decaf::internal::net::URIHelper, 3874	

---

- activemq::commands::DestinationInfo, 1695
- activemq::commands::FlushCommand, 1902
- activemq::commands::KeepAliveInfo, 2228
- activemq::commands::Message, 2490
- activemq::commands::MessageAck, 2525
- activemq::commands::MessageDispatchChannel, 2558
- activemq::commands::MessageDispatchNotification, 2594
- activemq::commands::MessagePull, 2699
- activemq::commands::ProducerAck, 2987
- activemq::commands::ProducerInfo, 3046
- activemq::commands::RemoveInfo, 3140
- activemq::commands::RemoveSubscriptionInfo, 3168
- activemq::commands::ReplayCommand, 3196
- activemq::commands::Response, 3230
- activemq::commands::SessionInfo, 3351
- activemq::commands::ShutdownInfo, 3415
- activemq::commands::TransactionInfo, 3788
- activemq::commands::WireFormatInfo, 3922
- voidp
  - zconf.h, 4430
- voidpc
  - zconf.h, 4430
- voidpf
  - zconf.h, 4430
- w\_bits
  - internal\_state, 2084
- w\_mask
  - internal\_state, 2084
- w\_size
  - internal\_state, 2084
- wait
  - activemq::core::MessageDispatchChannel, 2565, 2566
  - decaf::internal::util::concurrent::Condition, 1229
  - decaf::internal::util::concurrent::SynchronizerImpl, 3657, 3658
  - decaf::io::InputStream, 2011, 2012
  - decaf::io::OutputStream, 2861, 2862
  - decaf::util::AbstractCollection, 159, 160
  - decaf::util::concurrent::ConcurrentStlMap, 1218, 1219
  - decaf::util::concurrent::Mutex, 2739, 2740
  - decaf::util::concurrent::Synchronizable, 3651–3653
  - decaf::util::StlMap, 3554–3556
  - decaf::util::StlQueue, 3563, 3564
  - WAIT\_INFINITE
    - Concurrent.h, 4511
  - waitForSpace
    - activemq::util::MemoryUsage, 2475
    - activemq::util::Usage, 3897
  - WAITING
    - decaf::lang::Thread, 3710
  - wakeup
    - activemq::core::ActiveMQSession, 502
    - activemq::core::ActiveMQSessionExecutor, 506
    - activemq::threads::CompositeTaskRunner, 1196
    - activemq::threads::DedicatedTaskRunner, 1640
    - activemq::threads::TaskRunner, 3681
  - want
    - gz\_state, 1941
  - Warn
    - decaf::util::logging, 144
  - warn
    - decaf::util::logging::SimpleLogger, 3445
  - WARNING
    - decaf::util::logging::Level, 2295
  - warning
    - decaf::util::logging::Logger, 2357
  - was
    - inflate\_state, 1984
  - wasPrepared
    - activemq::commands::JournalTransaction, 2201
  - wbits
    - inflate\_state, 1984
  - what
    - cms::CMSException, 1133
    - decaf::lang::Exception, 1800
  - wrap
    - inflate\_state, 1984
  - wrapInflate
    - deflate.h, 4420
  - window
    - inflate\_state, 1985
    - internal\_state, 2084

- window\_size
    - internal\_state, 2084
  - windowSize
    - activemq::commands::ProducerInfo, 3047
  - WireFormatInfo
    - activemq::commands::WireFormatInfo, 3914
  - WireFormatInfoMarshaller
    - activemq::wireformat::openwire::marshal::WireFormatInfoMarshaller, 3940
    - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3932
    - activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3944
    - activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3936
    - activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3924
    - activemq::wireformat::openwire::marshal::v6::WireFormatInfoMarshaller, 3928
  - WireFormatNegotiator
    - activemq::wireformat::WireFormatNegotiator, 3947
  - wnext
    - inflate\_state, 1985
  - work
    - inflate\_state, 1985
  - wrap
    - decaf::nio::ByteBuffer, 1022
    - decaf::nio::CharBuffer, 1106
    - decaf::nio::DoubleBuffer, 1785
    - decaf::nio::FloatBuffer, 1898, 1899
    - decaf::nio::IntBuffer, 2037, 2038
    - decaf::nio::LongBuffer, 2414, 2415
    - decaf::nio::ShortBuffer, 3412
    - inflate\_state, 1985
    - internal\_state, 2084
  - write
    - decaf::internal::net::ssl::openssl::OpenSSL::Socket, 2821
    - decaf::internal::net::tcp::TcpSocket, 3690
    - decaf::internal::util::ByteArrayAdapter, 950
    - decaf::io::OutputStream, 2863
    - decaf::io::Writer, 3955, 3956
  - WRITE\_FAILURE
    - decaf::util::logging::ErrorManager, 1794
  - writeBoolean
    - activemq::commands::ActiveMQBytesMessage, 214
  - activemq::commands::ActiveMQStreamMessage, 518
  - activemq::wireformat::openwire::utils::BooleanStream, 820
  - cms::BytesMessage, 1035
  - cms::StreamMessage, 3604
  - decaf::io::DataOutput, 1542
  - decaf::io::DataOutputStream, 1549
  - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3932
  - activemq::commands::ActiveMQBytesMessage, 215
  - activemq::commands::ActiveMQStreamMessage, 519
  - cms::BytesMessage, 1035
  - cms::StreamMessage, 3605
  - decaf::io::DataOutput, 1543
  - decaf::io::DataOutputStream, 1549
  - writeBytes
    - activemq::commands::ActiveMQBytesMessage, 215, 216
  - activemq::commands::ActiveMQStreamMessage, 519
  - cms::BytesMessage, 1036
  - cms::StreamMessage, 3605, 3606
  - decaf::io::DataOutput, 1543
  - decaf::io::DataOutputStream, 1549
  - writeChar
    - activemq::commands::ActiveMQBytesMessage, 216
  - activemq::commands::ActiveMQStreamMessage, 519
  - cms::BytesMessage, 1037
  - cms::StreamMessage, 3606
  - decaf::io::DataOutput, 1543
  - decaf::io::DataOutputStream, 1549
  - writeChars
    - decaf::io::DataOutput, 1544
    - decaf::io::DataOutputStream, 1549
  - decaf::internal::net::ssl::openssl::OpenSSL::Socket, 2821
  - activemq::transport::inactivity::InactivityMonitor, 1967
  - decaf::internal::net::tcp::TcpSocket, 3690
  - activemq::transport::inactivity::WriteChecker, 3951
  - decaf::io::OutputStream, 2863
  - writeDouble
    - activemq::commands::ActiveMQBytesMessage, 217
  - activemq::commands::ActiveMQStreamMessage, 520
  - cms::BytesMessage, 1037
  - cms::StreamMessage, 3607

decaf::io::DataOutput, 1544  
 decaf::io::DataOutputStream, 1549  
 writeFloat  
   activemq::commands::ActiveMQBytesMessage, 217  
   activemq::commands::ActiveMQStreamMessage, 520  
 cms::BytesMessage, 1037  
 cms::StreamMessage, 3607  
 decaf::io::DataOutput, 1544  
 decaf::io::DataOutputStream, 1549  
 writeInt  
   activemq::commands::ActiveMQBytesMessage, 217  
   activemq::commands::ActiveMQStreamMessage, 521  
 cms::BytesMessage, 1038  
 cms::StreamMessage, 3607  
 decaf::io::DataOutput, 1545  
 decaf::io::DataOutputStream, 1549  
 writeLock  
   decaf::util::concurrent::locks::ReadWriteLock, 3119  
 writeLong  
   activemq::commands::ActiveMQBytesMessage, 218  
   activemq::commands::ActiveMQStreamMessage, 521  
 cms::BytesMessage, 1038  
 cms::StreamMessage, 3608  
 decaf::io::DataOutput, 1545  
 decaf::io::DataOutputStream, 1549  
 Writer  
   decaf::io::Writer, 3952  
 writeShort  
   activemq::commands::ActiveMQBytesMessage, 218  
   activemq::commands::ActiveMQStreamMessage, 521  
 cms::BytesMessage, 1039  
 cms::StreamMessage, 3608  
 decaf::io::DataOutput, 1545  
 decaf::io::DataOutputStream, 1549  
 writeString  
   activemq::commands::ActiveMQBytesMessage, 219  
   activemq::commands::ActiveMQStreamMessage, 522  
   activemq::util::MarshallingSupport, 2454  
   cms::BytesMessage, 1039  
   cms::StreamMessage, 3609  
   writeString16  
     activemq::util::MarshallingSupport, 2454  
     activemq::util::MarshallingSupport, 2455  
   decaf::io::ByteArrayOutputStream, 994  
   writeUnsignedShort  
     activemq::commands::ActiveMQBytesMessage, 219  
     activemq::commands::ActiveMQStreamMessage, 522  
   cms::BytesMessage, 1039  
   cms::StreamMessage, 3609  
   decaf::io::DataOutput, 1546  
   decaf::io::DataOutputStream, 1549  
   writeUTF  
     activemq::commands::ActiveMQBytesMessage, 219  
     cms::BytesMessage, 1040  
     decaf::io::DataOutput, 1546  
     decaf::io::DataOutputStream, 1550  
     written  
       decaf::io::DataOutputStream, 1550  
   inflate\_state, 1985  
   XATransactionId  
     activemq::commands::XATransactionId, 3961  
   XATransactionIdMarshaller  
     activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3977  
     activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3969  
     activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3981  
     activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3973  
     activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3985  
     activemq::wireformat::openwire::marshal::v6::XATransactionIdMarshaller, 3965  
   xflags  
   xz\_header\_s, 1939  
   XMLFormatter  
     decaf::util::logging::XMLFormatter, 3989  
   yield  
   decaf::lang::Thread, 3715



- Z\_ASCII
  - zlib.h, 4433
- Z\_BEST\_COMPRESSION
  - zlib.h, 4433
- Z\_BEST\_SPEED
  - zlib.h, 4433
- Z\_BINARY
  - zlib.h, 4433
- Z\_BLOCK
  - zlib.h, 4433
- Z\_BUF\_ERROR
  - zlib.h, 4433
- Z\_DATA\_ERROR
  - zlib.h, 4433
- Z\_DEFAULT\_COMPRESSION
  - zlib.h, 4433
- Z\_DEFAULT\_STRATEGY
  - zlib.h, 4433
- Z\_DEFLATED
  - zlib.h, 4433
- z\_errmsg
  - zutil.h, 4440
- Z\_ERRNO
  - zlib.h, 4433
- Z\_FILTERED
  - zlib.h, 4433
- Z\_FINISH
  - zlib.h, 4433
- Z\_FIXED
  - zlib.h, 4434
- Z\_FULL\_FLUSH
  - zlib.h, 4434
- Z\_HUFFMAN\_ONLY
  - zlib.h, 4434
- Z\_MEM\_ERROR
  - zlib.h, 4434
- Z\_NEED\_DICT
  - zlib.h, 4434
- Z\_NO\_COMPRESSION
  - zlib.h, 4434
- Z\_NO\_FLUSH
  - zlib.h, 4434
- Z\_NULL
  - zlib.h, 4434
- z\_off64\_t
  - zconf.h, 4429
- z\_off\_t
  - zconf.h, 4429
- Z\_OK
  - zlib.h, 4434
- Z\_PARTIAL\_FLUSH
  - zlib.h, 4434
- Z\_RLE
  - zlib.h, 4434
- z\_stream
  - zlib.h, 4435
- Z\_STREAM\_END
  - zlib.h, 4434
- Z\_STREAM\_ERROR
  - zlib.h, 4434
- z\_stream\_s, 3990
  - adler, 3991
  - avail\_in, 3991
  - avail\_out, 3991
  - data\_type, 3991
  - msg, 3991
  - next\_in, 3991
  - next\_out, 3991
  - opaque, 3991
  - reserved, 3991
  - state, 3991
  - total\_in, 3991
  - total\_out, 3991
  - zalloc, 3991
  - zfree, 3991
- z\_streamp
  - zlib.h, 4435
- Z\_SYNC\_FLUSH
  - zlib.h, 4434
- Z\_TEXT
  - zlib.h, 4434
- Z\_TREES
  - zlib.h, 4434
- Z\_UNKNOWN
  - zlib.h, 4434
- Z\_VERSION\_ERROR
  - zlib.h, 4434
- ZALLOC
  - zutil.h, 4439
- zalloc
  - z\_stream\_s, 3991
- zconf.h
  - Byte, 4429
  - Bytef, 4429
  - charf, 4429
  - const, 4429
  - FAR, 4429
  - intf, 4429
  - MAX\_MEM\_LEVEL, 4429
  - MAX\_WBITS, 4429

- OF, 4429
- SEEK\_CUR, 4429
- SEEK\_END, 4429
- SEEK\_SET, 4429
- uInt, 4429
- uIntf, 4430
- uLong, 4430
- uLongf, 4430
- voidp, 4430
- voidpc, 4430
- voidpf, 4430
- z\_off64\_t, 4429
- z\_off\_t, 4429
- ZEXPORT, 4429
- ZEXPORTVA, 4429
- ZEXTERN, 4429
- ZEXPORT
  - zconf.h, 4429
- ZEXPORTVA
  - zconf.h, 4429
- ZEXTERN
  - zconf.h, 4429
- ZFREE
  - zutil.h, 4439
- zfree
  - z\_stream\_s, 3991
- ZipException
  - decaf::util::zip::ZipException, 3992, 3993
- zlib.h
  - deflateInit, 4433
  - deflateInit2, 4433
  - gz\_header, 4435
  - gz\_headerp, 4435
  - gzFile, 4435
  - inflateBackInit, 4433
  - inflateInit, 4433
  - inflateInit2, 4433
  - OF, 4435–4437
  - Z\_ASCII, 4433
  - Z\_BEST\_COMPRESSION, 4433
  - Z\_BEST\_SPEED, 4433
  - Z\_BINARY, 4433
  - Z\_BLOCK, 4433
  - Z\_BUF\_ERROR, 4433
  - Z\_DATA\_ERROR, 4433
  - Z\_DEFAULT\_COMPRESSION, 4433
  - Z\_DEFAULT\_STRATEGY, 4433
  - Z\_DEFLATED, 4433
  - Z\_ERRNO, 4433
  - Z\_FILTERED, 4433
  - Z\_FINISH, 4433
  - Z\_FIXED, 4434
  - Z\_FULL\_FLUSH, 4434
  - Z\_HUFFMAN\_ONLY, 4434
  - Z\_MEM\_ERROR, 4434
  - Z\_NEED\_DICT, 4434
  - Z\_NO\_COMPRESSION, 4434
  - Z\_NO\_FLUSH, 4434
  - Z\_NULL, 4434
  - Z\_OK, 4434
  - Z\_PARTIAL\_FLUSH, 4434
  - Z\_RLE, 4434
  - z\_stream, 4435
  - Z\_STREAM\_END, 4434
  - Z\_STREAM\_ERROR, 4434
  - z\_streamp, 4435
  - Z\_SYNC\_FLUSH, 4434
  - Z\_TEXT, 4434
  - Z\_TREES, 4434
  - Z\_UNKNOWN, 4434
  - Z\_VERSION\_ERROR, 4434
  - ZLIB\_VER\_MAJOR, 4434
  - ZLIB\_VER\_MINOR, 4434
  - ZLIB\_VER\_REVISION, 4434
  - ZLIB\_VER\_SUBREVISION, 4434
  - ZLIB\_VERNUM, 4434
  - ZLIB\_VERSION, 4435
  - zlib\_version, 4434
  - ZLIB\_INTERNAL
    - gzguts.h, 4422
    - zutil.h, 4439
  - ZLIB\_VER\_MAJOR
    - zlib.h, 4434
  - ZLIB\_VER\_MINOR
    - zlib.h, 4434
  - ZLIB\_VER\_REVISION
    - zlib.h, 4434
  - ZLIB\_VER\_SUBREVISION
    - zlib.h, 4434
  - ZLIB\_VERNUM
    - zlib.h, 4434
  - ZLIB\_VERSION
    - zlib.h, 4435
  - zlib\_version
    - zlib.h, 4434
  - zstrerror
    - gzguts.h, 4422
  - zutil.h
    - Assert, 4438
    - DEF\_MEM\_LEVEL, 4438

DEF\_WBITS, 4438  
DYN\_TREES, 4438  
ERR\_MSG, 4438  
ERR\_RETURN, 4439  
F\_OPEN, 4439  
local, 4439  
MAX\_MATCH, 4439  
MIN\_MATCH, 4439  
OF, 4440  
OS\_CODE, 4439  
PRESET\_DICT, 4439  
STATIC\_TREES, 4439  
STORED\_BLOCK, 4439  
Trace, 4439  
Tracec, 4439  
Tracecv, 4439  
Tracev, 4439  
Tracevv, 4439  
TRY\_FREE, 4439  
uch, 4439  
uchf, 4439  
ulg, 4439  
ush, 4439  
ushf, 4440  
z\_errmsg, 4440  
ZALLOC, 4439  
ZFREE, 4439  
ZLIB\_INTERNAL, 4439